K 1

```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
```

```
_$2



Val
---
000
000
000
7FI
7FI
7FI
7FI
7FI
7FI
7FI
7FI
7FI
```

```
**FILE**ID**UETDR1W00                              K  4
```

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR       11       WW      WW    000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR       11       WW      WW    000000      000000
UU      UU  EE              TT      DD      DD  RR      RR   1111       WW      WW  00      00  00      00
UU      UU  EE              TT      DD      DD  RR      RR   1111       WW      WW  00      00  00      00
UU      UU  EE              TT      DD      DD  RR      RR     11       WW      WW  00    0000  00    0000
UU      UU  EE              TT      DD      DD  RR      RR     11       WW      WW  00    0000  00    0000
UU      UU  EEEEEEEE        TT      DD      DD  RRRRRRRR       11       WW      WW  00  00  00  00  00  00
UU      UU  EEEEEEEE        TT      DD      DD  RRRRRRRR       11       WW      WW  00  00  00  00  00  00
UU      UU  EE              TT      DD      DD  RR  RR         11       WW  WW  WW  0000    00  0000    00
UU      UU  EE              TT      DD      DD  RR  RR         11       WW  WW  WW  0000    00  0000    00
UU      UU  EE              TT      DD      DD  RR    RR       11       WWWW  WWWW  00      00  00      00    ....
UU      UU  EE              TT      DD      DD  RR    RR       11       WWWW  WWWW  00      00  00      00    ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD    RR      RR   111111     WW      WW    000000      000000      ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD    RR      RR   111111     WW      WW    000000      000000      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
UET
V04
```

Table of contents

```
0000             .TITLE  UETDR1W00 - VAX/VMS UETP DR11-W EXERCISER
0000             .IDENT  'V04-000'
0000             .ENABLE SUPPRESSION
0000     ;
0000     ;****************************************************************************
0000     ;*                                                                          *
0000     ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     ;*  ALL RIGHTS RESERVED.                                                    *
0000     ;*                                                                          *
0000     ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0000     ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000     ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0000     ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0000     ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
0000     ;*  TRANSFERRED.                                                            *
0000     ;*                                                                          *
0000     ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0000     ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000     ;*  CORPORATION.                                                            *
0000     ;*                                                                          *
0000     ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0000     ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000     ;*                                                                          *
0000     ;*                                                                          *
0000     ;****************************************************************************
0000     ;
0000
0000     ;++
0000     ; FACILITY:
0000     ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000     ;       account.
0000     ;
0000     ; ABSTRACT:
0000     ;       Using $QIO System Services, this program exercises the maintenance mode
0000     ;       functions of a DR11-W.
0000     ;
0000     ; ENVIRONMENT:
0000     ;       This program will run in user access mode, with ASTs enabled except
0000     ;       during error processing.  The program requires an AST limit of 6, a
0000     ;       buffered I/O limit of 10(10) and the PHY_IO and DIAGNOSE privileges.
0000     ;
0000     ;--
0000     ;
0000     ; AUTHOR: Richard N. Holstein, CREATION DATE: August, 1981
0000     ;
0000     ; MODIFIED BY:
0000     ;
0000     ;       V03-005 RNH0006         Richard N. Holstein,    15-Feb-1984
0000     ;               Take advantage of the new UETP message codes.  Fix SSERROR
0000     ;               interaction with RMS_ERROR.
0000     ;
0000     ;       V03-004 RNH0005         Richard N. Holstein,    19-Dec-1983
0000     ;               Give correct sentinels to Test Controller.  Use LIB$SIGNAL or
0000     ;               $PUTMSG throughout, instead of LIB$PUT_OUTPUT.
0000     ;
0000     ;       V03-003 RNH0004         Richard N. Holstein,    11-Mar-1983
```

```
0000  ;                     Don't signal ending message in EXIT_HANDLER.
0000  ;
0000  ;           V03-002 RNH0003         Richard N. Holstein,    23-Feb-1983
0000  ;                   Allow for longer device names.
0000  ;
0000  ;           V03-001 RNH0002         Richard N. Holstein,    15-Oct-1982
0000  ;                   Miscellaneous fixes listed in the V3B UETP Workplan.
0000  ;
0000  ;           V02-001 RNH0001         Richard N. Holstein,    4-Dec-1981
0000  ;                   Fix problems in reset routine which caused errors when the
0000  ;                   turnaround connector wasn't installed.
0000  ;
0000  ;**
```

```
                     0000           .SBTTL  Declarations
                     0000   ;
                     0000   ; INCLUDE FILES:
                     0000   ;
                     0000   ;       SYS$LIBRARY:LIB.MLB     for general definitions
                     0000   ;       SHRLIB$:UETP.MLB        for UETP definitions
                     0000   ;
                     0000   ; MACROS:
                     0000   ;
                     0000           $CHFDEF                                 ; Condition handler frame definitions
                     0000           $DEVDEF                                 ; Device definitions
                     0000           $DIBDEF                                 ; Device Information Block
                     0000           $DVIDEF                                 ; $GETDVI ITMLST item codes
                     0000           $IODEF                                  ; I/O functions codes, etc.
                     0000           $QIODEF                                 ; $QIO offsets and NARGS
                     0000           $SHRDEF                                 ; Shared messages
                     0000           $SSDEF                                  ; System Service status codes
                     0000           $STSDEF                                 ; Status return
                     0000           $UETUNTDEF                              ; UETP unit block offset definitions
                     0000           $UETPDEF                                ; UETP
                     0000           $XADEF                                  ; DR11-W
                     0000   ;
                     0000   ; EQUATED SYMBOLS:
                     0000   ;
                     0000   ;   Facility number definitions:
          00000001   0000           RMS$_FACILITY = 1
                     0000
                     0000   ;   SHR message definitions:
          00740000   0000           UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
          007410E0   0000           UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
          00741038   0000           UETP$_BEGIND = UETP!SHR$_BEGIND
          00741080   0000           UETP$_ENDEDD = UETP!SHR$_ENDEDD
          00741098   0000           UETP$_OPENIN = UETP!SHR$_OPENIN
          00741130   0000           UETP$_TEXT   = UETP!SHR$_TEXT
                     0000
                     0000   ;   Internal flag bits...:
          00000001   0000           TEST_OVERV   = 1                        ; Set when test is over
          00000002   0000           SAFE_TO_UPDV = 2                        ; Set if it's safe to update UETINIDEV
          00000003   0000           BEGIN_MSGV   = 3                        ; Set if "BEGIN" msg has been printed
          00000004   0000           ONE_SHOTV    = 4                        ; Set if running in one-shot mode
          00000005   0000           DUMP_MODEV   = 5
          00000006   0000           NO_MESSAGEV  = 6                        ; Set if bad data msg given after $QIO
                     0000   ;   ...and corresponding masks:
          00000002   0000           TEST_OVERM   = 1@TEST_OVERV
          00000004   0000           SAFE_TO_UPDM = 1@SAFE_TO_UPDV
          00000008   0000           BEGIN_MSGM   = 1@BEGIN_MSGV
          00000010   0000           ONE_SHOTM    = 1@ONE_SHOTV
          00000020   0000           DUMP_MODEM   = 1@DUMP_MODEV
          00000040   0000           NO_MESSAGEM  = 1@NO_MESSAGEV
                     0000
                     0000   ;   Miscellany:
          00000020   0000           LC_BITM       = ^X20                    ; Mask to convert lower case to upper
          00000028   0000           REC_SIZE      = 40                      ; UETINIDEV.DAT record size
          000001F4   0000           TEXT_BUFFER   = 500                     ; Internal text buffer size
          00000004   0000           EFN2          = 4                       ; EFN used for three minute timer
          00000003   0000           SS_SYNCH_EFN  = 3                       ; Synch miscellaneous system services
          0000000F   0000           MAX_PROC_NAME = 15                      ; Longest possible process name
```

```
                    0000000A  0000    129          MAX_DEV_DESIG = 10                     ; Longest possible controller name
                    00000005  0000    130          MAX_UNIT_DESIG= 5                      ; Longest possible unit number
```

```
          0000  ;   DR11-W specific definitions:
00000005  0000          QIO_EFN  = EFN2+1                  ; EFN for DR11-W I/O
000003E8  0000          DWT_SIZE = 1000                    ; Typical DR11-W transfer size in bytes
          0000  .IIF NE DWT_SIZE&1, .ERROR DWT_SIZE       ; DWT_SIZE must be an even number!
          0000  .IIF GE DWT_SIZE-65535, .ERROR DWT_SIZE   ; DWT_SIZE must be less than 65535!
00000258  0000          MINIMUM  = 300000/<DWT_SIZE/2>     ; Min. acceptable $QIOs for normal run
          0000          ;           = transfer-rate-in-words-per-second/transfer-size-in-words
          0000          ; The test lasts well over a second and not all $QIOs transfer so many
          0000          ; bytes, so this should be quite a minimal value!
          0000
          0000  ; Note well that the DR11-W transfers words, but VMS counts bytes, and that in
          0000  ; maintenance mode, "the DR11-W does alternating DATI/DATO transfers at
          0000  ; consecutive locations; a DATI from location X followed by a DATO to location
          0000  ; X+2, followed by a DATI from location X+4 and so on." - DR11-W Specification,
          0000  ; September 1980 revision.  The word count is decremented twice for each word
          0000  ; of data, once for the DATI and once for the DATO.
000003E8  0000          WRITE_SIZE = DWT_SIZE              ; Buffer size in bytes
          0000
          0000  ; For conciseness later on, define here bit masks for I/O function codes.
0000190B  0000          XAW__RESET_CYCLE   = IO$_WRITEPBLK!IO$M_DIAGNOSTIC!IO$M_RESET!IO$M_CYCLE
0000110B  0000          XAW__CYCLE         = IO$_WRITEPBLK!IO$M_DIAGNOSTIC!IO$M_CYCLE
0000130B  0000          XAW__SETFNCT_CYCLE = IO$_WRITEPBLK!IO$M_DIAGNOSTIC!IO$M_SETFNCT!IO$M_CYCLE
0000118B  0000          XAW__TIMED_CYCLE   = IO$_WRITEPBLK!IO$M_DIAGNOSTIC!IO$M_TIMED!IO$M_CYCLE
          0000
          0000  ;
          0000  ; For each unit, there will be a data structure set up, called a
          0000  ; node.  These nodes will be linked together in a self-relative queue whose
          0000  ; header is UNIT_LIST.  The first part of each node will be the standard
          0000  ; definition from $UETUNTDEF.  Following that will come the device test
          0000  ; dependent stuff, defined below.  NOTE THAT THIS DEFINITION IS DONE WITH AN
          0000  ; ABSOLUTE PSECT.  This means that what look like declarations are really
          0000  ; definitions and the labels are really just offsets into a given node on the
          0000  ; queue.  (A not necessarily obvious consequence of using an ABS PSECT is that
          0000  ; space must be reserved with .BLKx operations, since .BYTE, etc., attempt to
          0000  ; store data.)
          0000  ;
          0000          .PSECT  DEVDEP_STR_DEF,ABS,NOEXE,NOWRT,PAGE ; Note ABS attribute!
          0000
000001A4  0000          .BLKB   UETUNT$C_DEVDEP            ; Skip over standard UETUNT block
          01A4
          01A4  XA_Q_IOSB:                                 ; IOSB for our DR11-W
000001AC  01A4          .BLKQ   1
          01AC
          01AC  XA_Q_CHARAC:                               ; Characteristics buffer for IO$_SETCHAR
000001B4  01AC          .BLKQ   1
          01B4
          01B4  XA_Q_ORIGINAL:                             ; DR11-W characteristics before starting tes
000001BC  01B4          .BLKQ   1
          01BC
          01BC  XA_K_QIO:                                  ; Space for $QIO_G argument list...
000001F0  01BC          .BLKL   QIO$_NARGS+1               ; ...and the argument list counter
          01F0
          01F0  ;
          01F0  ; All the code which uses the following two items is heavily dependent on its
          01F0  ; dealing with quadword pairs (i.e., 16-byte) of time stamps and on the
          01F0  ; number of quadword pairs fitting into a byte.
          01F0  ;
```

```
00000100  01F0          TIME_STAMP_LEN = 256               ; 2*count of start/finish quadword pairs
          01F0      XA_B_TSI:                              ; Index into the following...
000001F1  01F0              .BLKB   1
          01F1      XA_K_TSTAMP:                           ; ...circular list of time stamps of...
000009F1  01F1              .BLKQ   TIME_STAMP_LEN         ; ...last start/finish of $QIOs
          09F1
0000084D  09F1          DEVDEP_SIZE = .-UETUNT$C_DEVDEP    ; Device dependent part size of node
          09F1                                             ; Note that this excludes buffers
          09F1
          09F1              .ALIGN  WORD                   ; Unibus DDP requires word alignment
          09F2      XA_K_BUF:                              ; I/O buffer for both reads and writes
00000DDA  09F2              .BLKB   WRITE_SIZE
          0DDA
          0DDA          PAGES = <<UETUNT$C_INDSIZ+-        ; Add together all of the pieces...
          0DDA                  DEVDEP_SIZE+-              ; ...which make up a UETP unit block...
          0DDA                  WRITE_SIZE+-               ; ...to give to the $EXPREG service...
00000007  0DDA                  511>/512>                  ; ...later
```

```
                                      0DDA          .SBTTL  Read-Only Data
                                  00000000          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                      0000
                                      0000  ACNT_NAME:                                      ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'  0000          .ASCID  /SYSTEST/
                                  54  000E
                                      000F
                                      000F  TEST_NAME:                                      ; This test name
31 52 44 54 45 55 00000017'010E0000'  000F          .ASCID  /UETDR1W00/
                            30 30 57  001D
                                      0020
                                      0020  SUPDEV_GBLSEC:                                  ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'  0020          .ASCID  /UETSUPDEV/
                            56 45 44  002E
                                      0031
                                      0031  CONTROLLER:                                     ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'  0031          .ASCID  /CTRLNAME/
                               45 4D  003F
                                      0041
                                      0041  MODE:                                           ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'  0041          .ASCID  /MODE/
                                      004D
                                      004D  NO_RMS_AST_TABLE:                               ; List of errors for which...
                           00000000'  004D          .LONG   RMS$_BLN                        ; ...RMS cannot deliver an AST...
                           00000000'  0051          .LONG   RMS$_BUSY                       ; ...even if one has an ERR= arg
                           00000000'  0055          .LONG   RMS$_CDA                        ; Note that we can search table...
                           00000000'  0059          .LONG   RMS$_FAB                        ; ...via MATCHC since <31:16>...
                           00000000'  005D          .LONG   RMS$_RAB                        ; ...pattern can't be in <15:0>
                           00000014   0061  NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                      0061
                                      0061  SYS$INPUT:                                      ; Name of device from which...
4E 49 24 53 59 53 00000069'010E0000'  0061          .ASCID  /SYS$INPUT/                     ; ...the test can be aborted
                            54 55 50  006F
                                      0072
                                      0072  INPUT_ITMLST:                                   ; $GETDVI arg list for SYS$INPUT
                           0020 0040  0072          .WORD   64,DVI$_DEVNAM                  ; We need the equivalence name
                  0000000C'00000014'  0076          .LONG   BUFFER,BUFFER_PTR
                           00000000   007E          .LONG   0                               ; Terminate the list
                                      0082
                                      0082  CS1:                                            ; Device class and type control string
21 20 42 58 32 21 0000008A'010E0000'  0082          .ASCID  /!2XB !2XB /
                         20 42 58 32  0090
                                      0094
                                      0094  CS3:                                            ; Device class-only control string
2A 20 42 58 32 21 0000009C'010E0000'  0094          .ASCID  /!2XB **/
                                  2A  00A2
                                      00A3
                                      00A3  CNTRLCMSG:
65 74 72 6F 62 41 000000AB'010E0000'  00A3          .ASCID  \Aborted via a user CTRL/C\
 72 65 73 75 20 61 20 61 69 76 20 64  00B1
                43 2F 4C 52 54 43 20  00BD
                                      00C4
                                      00C4  NO_CTRLNAME:
6E 6F 63 20 6F 4E 000000CC'010E0000'  00C4          .ASCID  /No controller specified./
 63 65 70 73 20 72 65 6C 6C 6F 72 74  00D2
                   2E 64 65 69 66 69  00DE
                                      00E4
                                      00E4  253 DEAD_CTRLNAME:
20 74 27 6E 61 43 000000EC'010E0000'  00E4  254         .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74   00F2
72 61 6D 20 2C 53 41 21 20 72 65 6C   00FE
61 73 75 6E 75 20 73 61 20 64 65 6B   010A
4E 49 54 45 55 20 6E 69 20 65 6C 62   0116
         2E 54 41 44 2E 56 45 44 49   0122
                                      012B  255
                                      012B  256 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000133'010E0000'  012B  257         .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74   0139
2E 67 6E 69 74 73 65 74 20 72 6F 66   0145
                                      0151  258
                                      0151  259 ILLEGAL_REC:
61 67 65 6C 6C 49 00000159'010E0000'  0151  260         .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C   015F
20 65 6C 69 66 20 6E 69 20 74 61 6D   016B
41 44 2E 56 45 44 49 4E 49 54 45 55   0177
                                21 54   0183
                                      0185  261
                                      0185  262 PASS_MSG:
66 6F 20 64 6E 45 0000018D'010E0000'  0185  263         .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20   0193
61 72 65 74 69 20 4C 55 21 20 68 74   019F
44 25 21 20 74 61 20 73 6E 6F 69 74   01AB
                                   2E   01B7
                                      01B8  264
                                      01B8  265 INIDEV_UPDERR:                          ; Error during exit handler
20 72 6F 72 72 45 000001C0'010E0000'  01B8  266         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75   01C6
   2E 54 41 44 2E 56 45 44 49 4E 49   01D2
                                      01DD  267
                                      01DD  268 THREEMIN:                               ; 3 minute delta time
                    FFFFFFFF 94B62E00  01DD  269         .LONG   -10*1000*1000*180,-1
                                      01E5  270
                                      01E5  271 UNIT_DESC:                              ; Descriptor used to convert unit #
                             00000005  01E5  272         .LONG   5
                            0000001A'  01E9  273         .ADDRESS BUFFER+6
                                      01ED  274
                                      01ED  275 CONT_DESC:                              ; Descriptor used to convert controller...
                            0000 0028  01ED  276         .WORD   REC_SIZE,0              ; ...from lowercase to uppercase
                            00000014'  01F1  277         .ADDRESS BUFFER
                                      01F5  278
                                      01F5  279 FILE:                                   ; Fills in RMS_ERR_STRING
      65 6C 69 66 000001FD'010E0000'  01F5  280         .ASCID  /file/
                                      0201  281
                                      0201  282 RECORD:                                 ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 00000209'010E0000'  0201  283         .ASCID  /record/
                                      020F  284
                                      020F  285 RMS_ERR_STRING:                         ; Announces an RMS error
41 21 20 53 4D 52 00000217'010E0000'  020F  286         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53   021D
         44 41 21 20 65 6C 69          0229
                                      0230  287
                                      0230  288 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43   0230  289         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65   023C
```

```
                                   20  0248
                             00000019  0249   290           PMTSIZ = .-PROMPT
                                       0249   291
                                       0249   292   DEVALLOC:                              ; Warns if DR11-W already assigned
73 69 20 53 41 21 00000251'010E0000'   0249   293           .ASCID  /!AS is not available for testing./
62 61 6C 69 61 76 61 20 74 6F 6E 20    0257
69 74 73 65 74 20 72 6F 66 20 65 6C    0263
                            2E 67 6E   026F
                                       0272   294
                                       0272   295   SLOW_DR11W:                            ; Warns if DR11-W didn't do min. I/O
6E 6F 20 43 41 21 0000027A'010E0000'   0272   296           .ASCID  /!AC only completed !UL $QIOs./
64 65 74 65 6C 70 6D 6F 63 20 79 6C    0280
   2E 73 4F 49 51 24 20 4C 55 21 20    028C
                                       0297   297
                                       0297   298   QIO_ERROR:                             ; Message if $QIO itself fails
65 20 4F 49 51 24 0000029F'010E0000'   0297   299           .ASCID  /$QIO error on device !AC./
69 76 65 64 20 6E 6F 20 72 6F 72 72    02A5
                  2E 43 41 21 20 65 63 02B1
                                       02B8   300
                                       02B8   301   QIO_FUNC_FAIL:                         ; Message if $QIO function failed
6F 20 4F 49 51 24 000002C0'010E0000'   02B8   302           .ASCID  \$QIO on !AC failed, function !XL, IOSB !XL !XL.\
64 65 6C 69 61 66 20 43 41 21 20 6E    02C6
21 20 6E 6F 69 74 63 6E 75 66 20 2C    02D2
4C 58 21 20 42 53 4F 49 20 2C 4C 58    02DE
                     2E 4C 58 21 20    02EA
                                       02EF   303
                                       02EF   304   DEBUG_MSG:                             ; Useful stuff if test dies
75 62 65 44 2F 21 000002F7'010E0000'   02EF   305           .ASCID  \!/Debug info:  R0 = !XL, QIO$_EFN(R10) = !XL,\-
20 30 52 20 20 3A 6F 66 6E 69 20 67    02FD
5F 24 4F 49 51 20 2C 4C 58 21 20 3D    0309
21 20 3D 20 29 30 31 52 28 4E 46 45    0315
                            2C 4C 58   0321
28 4E 41 48 43 5F 24 4F 49 51 2F 21    0324   306                   \!/QIO$_CHAN(R10) = !XL, QIO$_FUNC(R10) = !XL,\-
20 2C 4C 58 21 20 3D 20 29 30 31 52    0330
31 52 28 43 4E 55 46 5F 24 4F 49 51    033C
            2C 4C 58 21 20 3D 20 29 30 0348
44 41 54 53 41 5F 24 4F 49 51 2F 21    0351   307                   \!/QIO$_ASTADR(R10) = !XL, QIO$_ASTPRM(R10) = !XL,\-
4C 58 21 20 3D 20 29 30 31 52 28 52    035D
52 50 54 53 41 5F 24 4F 49 51 20 2C    0369
4C 58 21 20 3D 20 29 30 31 52 28 4D    0375
                                  2C   0381
31 52 28 31 50 5F 24 4F 49 51 2F 21    0382   308                   \!/QIO$_P1(R10) = !XL, QIO$_P2(R10) = !XL,\-
49 51 20 2C 4C 58 21 20 3D 20 29 30    038E
3D 20 29 30 31 52 28 32 50 5F 24 4F    039A
                     2C 4C 58 21 20    03A6
31 52 28 33 50 5F 24 4F 49 51 2F 21    03AB   309                   \!/QIO$_P3(R10) = !XL, QIO$_P4(R10) = !XL, QIO$_P5(R10) = !XL,\-
49 51 20 2C 4C 58 21 20 3D 20 29 30    03B7
3D 20 29 30 31 52 28 34 50 5F 24 4F    03C3
50 5F 24 4F 49 51 20 2C 4C 58 21 20    03CF
4C 58 21 20 3D 20 29 30 31 52 28 35    03DB
                                  2C   03E7
28 42 53 4F 49 5F 51 5F 41 58 2F 21    03E8   310                   \!/XA_Q_IOSB(R6) = !XL, XA_Q_IOSB+4(R6) = !XL\-
58 20 2C 4C 58 21 20 3D 20 29 36 52    03F4
52 28 34 2B 42 53 4F 49 5F 51 5F 41    0400
               4C 58 21 20 3D 20 29 36 040C
41 52 41 48 43 5F 51 5F 41 58 2F 21    0414   311                   \!/XA_Q_CHARAC(R6) = !XL, XA_Q_CHARAC+4(R6) = !XL\
2C 4C 58 21 20 3D 20 29 36 52 28 43    0420
```

```
43 41 52 41 48 43 5F 51 5F 41 58 20  042C
4C 58 21 20 3D 20 29 36 52 28 34 2B  0438
                                     0444   312
                                     0444   313 DIAG_MSG:                               ; Contents of $QIO P6 buffer
67 61 69 44 2F 21 0000044C'010E0000' 0444   314         .ASCID  \!/Diagnostic buffer: !6(9XL)!/\-
65 66 66 75 62 20 63 69 74 73 6F 6E  0452
2F 21 29 4C 58 39 28 36 21 20 3A 72  045E
21 20 3D 20 50 4D 54 52 53 43 2F 21  046A   315                 \!/CSRTMP = !XL, BARTMP = !XL, CSR    = !XL, EIR    = !XL\-
3D 20 50 4D 54 52 41 42 20 2C 4C 58  0476
20 20 20 52 53 43 20 2C 4C 58 21 20  0482
20 52 49 45 20 2C 4C 58 21 20 3D 20  048E
            4C 58 21 20 3D 20 20 20  049A
21 20 3D 20 20 20 20 52 44 49 2F 21  04A2   316                 \!/IDR    = !XL, BAR    = !XL, WCR    = !XL, ERROR  = !XL\-
3D 20 20 20 20 52 41 42 20 2C 4C 58  04AE
20 20 20 52 43 57 20 2C 4C 58 21 20  04BA
4F 52 52 45 20 2C 4C 58 21 20 3D 20  04C6
            4C 58 21 20 3D 20 20 52  04D2
21 20 3D 20 6D 75 6E 52 50 44 2F 21  04DA   317                 \!/DPRnum = !XL, DPRcon = !XL, FMPR   = !XL, PMPR   = !XL\-
3D 20 6E 6F 63 52 50 44 20 2C 4C 58  04E6
20 20 52 50 4D 46 20 2C 4C 58 21 20  04F2
52 50 4D 50 20 2C 4C 58 21 20 3D 20  04FE
            4C 58 21 20 3D 20 20 20  050A
21 20 3D 20 72 61 70 52 50 44 2F 21  0512   318                 \!/DPRpar = !XL!/!8(9XL)\
   29 4C 58 39 28 38 21 2F 21 4C 58  051E
```

```
0529  ;+
0529  ; The TABLE_GEN macro is used to coherently and consistently lay out the
0529  ; parameters which will change from $QIO to $QIO when testing the DR11-W.
0529  ; Each line is a call to the LINE_GEN macro.  The LINE_GEN macro will be
0529  ; expanded to fill in a set of parallel tables from which the parameters
0529  ; will be taken when the $QIO is performed.
0529  ;
0529  ; Because these tables introduce an additional level of indirection in the
0529  ; arguments, the typical $QIO_S form of the system service becomes unuseable.
0529  ; We will use the $QIO_G form.  The argument list will be reserved space in
0529  ; the node on UNIT_LIST for the DR11-W; we can't use the $QIO macro there
0529  ; because of ABS .PSECT restrictions.  Define a dummy argument list now with
0529  ; the items which can be supplied at assembly time:
0529  ;
0529  DUMMY_QIO:
0529          $QIO    EFN = QIO_EFN, P6 = DIAG_BUF
055D  ;
055D  ; The rest of the arguments will be supplied as the node is allocated (those
055D  ; which are static across $QIOs) or as an individual $QIO is done.
055D  ;-
055D
055D  .MACRO  TABLE_GEN
055D  ;
055D  ; The function codes used below are all defined earlier in the DR11-W Specific
055D  ; definitions area.
055D  ;
055D  ; First, try some basic functions:  in maintenance mode, do various length word
055D  ; and block mode transfers.  Since logical, virtual and physical I/O are the
055D  ; same to the DR11-W, doing all I/O in physical mode is sufficient.  Physical
055D  ; mode I/O is necessary to access the DR11-W in maintenance mode.
055D  ;
055D          LINE_GEN IO$_SETCHAR, 0, XA_Q_CHARAC, 0,  0,  0,  0
055D  ;
055D  ;       LINE_GEN            FUNC, ASTADR,       P1,       P2, P3, P4, P5
055D          LINE_GEN XAW__RESET_CYCLE, 0, XA_K_BUF,        4,  0,  0,  0
055D          LINE_GEN XAW__RESET_CYCLE, 0, XA_K_BUF,        4,  0,  0,  0
055D          LINE_GEN       XAW__CYCLE, 0, XA_K_BUF,        4,  0,  0,  0
055D          LINE_GEN       XAW__CYCLE, 0, XA_K_BUF,DWT_SIZE,  0,  0,  0
055D          LINE_GEN       XAW__CYCLE, 0, XA_K_BUF,DWT_SIZE,  0,  0,  0
055D  ;
055D  ; Play with FNCT and STATUS bits.  The set won't transfer any
055D  ; data, per se, but will cause the DR11-W IDR and ODR to be accessed.
055D  ; NOTE: This function cancelled because it works only if the turnaround
055D  ; connector is installed.
055D  ;
055D  ;       LINE_GEN              FUNC, ASTADR,      P1, P2, P3, P4,      P5
055D  ;       LINE_GEN XAW__SETFNCT_CYCLE, 0, XA_K_BUF,  0,  0,  5, <^XA72E>
055D  ;
055D  ; Do some word and block mode transfers with a timeout parameter.  Get an
055D  ; AST when the device finishes.
055D  ;
055D  ;       LINE_GEN            FUNC, ASTADR,       P1,       P2, P3, P4, P5
055D          LINE_GEN XAW__TIMED_CYCLE, IOAST, XA_K_BUF, DWT_SIZE,  2,  0,  0
055D
055D  .ENDM   TABLE_GEN
```

```
          055D   ;+
          055D   ; We now need to generate the set of parallel tables from which the $QIOs
          055D   ; will take their arguments.  Define the LINE_GEN macro twice, the first time
          055D   ; to count the number of calls, and the second time to fill the tables.  In
          055D   ; between, allocate the space for the tables.
          055D   ;-
          055D                 .MACRO  LINE_GEN FUNC,ASTADR,P1,P2,P3,P4,P5
          055D                 LINE_GEN_COUNT = LINE_GEN_COUNT+1
          055D                 .ENDM   LINE_GEN
          055D
00000000  055D                 LINE_GEN_COUNT = 0
          055D                 TABLE_GEN                          ; This one counts LINE_GEN calls
          055D
          055D                 .ALIGN  LONG                       ; May as well speed things up a bit
          0560
          0560   FUNC_TABLE:                                      ; $QIO function code
0000057C  0560                 .BLKL   LINE_GEN_COUNT
          057C
          057C   ASTADR_TABLE:                                    ; AST routine when I/O completes
00000598  057C                 .BLKL   LINE_GEN_COUNT
          0598
          0598   P1_TABLE:                                        ; Data buffer, characteristics buffer
000005B4  0598                 .BLKL   LINE_GEN_COUNT             ; or attention AST service routine
          05B4
          05B4   P2_TABLE:                                        ; Byte size of data buffer
000005D0  05B4                 .BLKL   LINE_GEN_COUNT
          05D0
          05D0   P3_TABLE:                                        ; Timeout in seconds or AST access mode
000005EC  05D0                 .BLKL   LINE_GEN_COUNT
          05EC
          05EC   P4_TABLE:                                        ; CSR FNCT bits (2-0 only)
00000608  05EC                 .BLKL   LINE_GEN_COUNT
          0608
          0608   P5_TABLE:                                        ; Value (word) to load into ODR
00000624  0608                 .BLKL   LINE_GEN_COUNT
          0624
          0624                 .MACRO  LINE_GEN        FUNC,ASTADR,P1,P2,P3,P4,P5
          0624                 . = FUNC_TABLE+<4*LINE_GEN_COUNT>
          0624                 .LONG   FUNC
          0624                 . = ASTADR_TABLE+<4*LINE_GEN_COUNT>
          0624                 .ADDRESS ASTADR
          0624                 . = P1_TABLE+<4*LINE_GEN_COUNT>
          0624                 .ADDRESS P1
          0624                 . = P2_TABLE+<4*LINE_GEN_COUNT>
          0624                 .LONG   P2
          0624                 . = P3_TABLE+<4*LINE_GEN_COUNT>
          0624                 .LONG   P3
          0624                 . = P4_TABLE+<4*LINE_GEN_COUNT>
          0624                 .LONG   P4
          0624                 . = P5_TABLE+<4*LINE_GEN_COUNT>
          0624                 .LONG   P5
          0624                 LINE_GEN_COUNT = LINE_GEN_COUNT+1
          0624                 .ENDM   LINE_GEN
          0624
00000000  0624                 LINE_GEN_COUNT = 0
          0624                 TABLE_GEN                          ; This one fills the above tables
```

```
                                  0624        .SBTTL  Read/Write Data
                              00000000        .PSECT  RWDATA,WRT,NOEXE,PAGE
                                  0000
                                  0000  TTCHAN:                                  ; Channel associated with ctrl. term.
                          0000    0000          .WORD   0
                                  0002
                                  0002  FLAG:                                    ; Miscellaneous flag bits
                          0000    0002          .WORD   0                        ; (See Equated Symbols for definitions)
                                  0004
                                  0004  FAO_BUF:                                 ; FAO output string descriptor
                     0000 01F4    0004          .WORD   TEXT_BUFFER,0
                     00000014'    0008          .ADDRESS BUFFER
                                  000C
                                  000C  BUFFER_PTR:                              ; Fake .ASCID buffer for misc. strings
                     0000 01F4    000C          .WORD   TEXT_BUFFER,0            ; A word for length, a word for desc.
                     00000014'    0010          .ADDRESS BUFFER
                                  0014
                                  0014  BUFFER:                                  ; FAO output and other misc. buffer
                      00000208    0014          .BLKB   TEXT_BUFFER
                                  0208
                                  0208  DEVDSC:                                  ; Device name descriptor
                     0000 000A    0208          .WORD   MAX_DEV_DESIG,0
                     00000227'    020C          .ADDRESS DEV_NAME
                                  0210
                                  0210  PROCESS_NAME:                            ; Process name
 57 31 52 44 00000218'010E0000'   0210          .ASCID  /DR1W/
                      0000000B    021C          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                      00000227    021C          .BLKB   PROCESS_NAME_FREE
                                  0227
                                  0227  DEV_NAME:                                ; Device name buffer
                      00000236    0227          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                      0000000F    0236          NAME_LEN = .-DEV_NAME
                                  0236
                                  0236  DIB:                                     ; Device Information Block
                     0000 0074    0236          .WORD   DIB$K_LENGTH,0
                     0000023E'    023A          .ADDRESS DIBBUF
                                  023E  DIBBUF:
                      000002B2    023E          .BLKB   DIB$K_LENGTH
                                  02B2
                                  02B2  ERROR_COUNT:                             ; Cumulative error count at runtime
                      00000000    02B2          .LONG   0
                                  02B6
                                  02B6  STATUS:                                  ; Status value on program exit
                      00000000    02B6          .LONG   0
                                  02BA
                                  02BA  QUAD_STATUS:                             ; IO status block for misc sys. svcs.
             00000000 00000000    02BA          .QUAD   0
                                  02C2
                                  02C2  INADDRESS:                               ; $CRMPSC address storage
             00000000 00000000    02C2          .LONG   0,0
                                  02CA
                                  02CA  OUTADDRESS:
             00000000 00000000    02CA          .LONG   0,0
                                  02D2
                                  02D2  DEVNAM_LEN:                              ; Current device name length
                          0000    02D2          .WORD   0
                                  02D4
```

```
                  02D4  ; RANDOM1 and RANDOM2 may be combined to produce a set of pseudo-random numbers
                  02D4  RANDOM1:                                      ; Random word #1
         AAAAAAAA 02D4          .LONG   ^XAAAAAAAA
                  02D8
                  02D8  RANDOM2:                                      ; Random word #2
         A72EA72E 02D8          .LONG   ^XA72EA72E
                  02DC
                  02DC  ITERATION:                                    ; # of times all tests were executed
         00000000 02DC          .LONG   0
                  02E0
                  02E0  PASS:                                         ; Pass count
         00000000 02E0          .LONG   0
                  02E4
                  02E4  MSG_BLOCK:                                    ; Auxiliary $GEYMSG info
         000002E8 02E4          .BLKB   4
                  02E8
                  02E8  EXIT_DESC:                                    ; Exit handler descriptor
         00000000 02E8          .LONG   0
        00000AC2' 02EC          .ADDRESS EXIT_HANDLER
         00000001 02F0          .LONG   1
        000002B6' 02F4          .ADDRESS STATUS
                  02F8
                  02F8  ARG_COUNT:                                    ; Argument counter used by ERROR_EXIT
         00000000 02F8          .LONG   0
                  02FC
                  02FC          .ALIGN QUAD                           ; For self-relative queue of unit blocks
                  0300
                  0300  UNIT_LIST:                                    ; Head of unit block circular list
00000000 00000000 0300          .QUAD   0
                  0308
                  0308  NEW_NODE:                                     ; Newly acquired node address
00000000 00000000 0308          .QUAD   0
                  0310
                  0310  DIAG_BUF:                                     ; $QIO P6 diagnostic buffer
         000003D8 0310          .BLKL   50
```

```
                 03D8          .SBTTL  RMS-32 Data Structures
                 03D8          .ALIGN  LONG
                 03D8
                 03D8   SYSIN_FAB:                                      ; Allocate FAB for SYS$INPUT
                 03D8          $FAB-
                 03D8          FNM = <SYS$INPUT>
                 0428
                 0428   SYSIN_RAB:                                      ; Allocate RAB for SYS$INPUT
                 0428          $RAB-
                 0428          FAB = SYSIN_FAB,-
                 0428          ROP = PMT,-
                 0428          PBF = PROMPT,-
                 0428          PSZ = PMTSIZ,-
                 0428          UBF = DEV_NAME,-
                 0428          USZ = NAME_LEN
                 046C
                 046C   INI_FAB:                                        ; Allocate FAB for UETINIDEV
                 046C          $FAB-
                 046C          FAC = <GET,PUT,UPD>,-
                 046C          RAT = CR,-
                 046C          SHR = <GET,PUT,UPI>,-
                 046C          FNM = <UETINIDEV.DAT>
                 04BC
                 04BC   INI_RAB:                                        ; Allocate RAB for UETINIDEV
                 04BC          $RAB-
                 04BC          FAB = INI_FAB,-
                 04BC          RBF = BUFFER,-
                 04BC          UBF = BUFFER,-
                 04BC          USZ = REC_SIZE
                 0500
                 0500   DDB_RFA:                                        ; RFA storage for INI_RAB
        00000506 0500          .BLKB   6
                 0506
                 0506          .ALIGN  LONG
                 0508   SUP_FAB:                                        ; Allocate FAB for UETSUPDEV
                 0508          $FAB-
                 0508          FAC = GET,-
                 0508          SHR = <UPI,GET>,-
                 0508          RAT = CR,-
                 0508          FOP = UFO,-
                 0508          FNM = <UETSUPDEV.DAT>
```

```
                              0558                .SBTTL  Test and Device Initialization
                          00000000                .PSECT  DR11W,EXE,NOWRT,PAGE
                              0000
                              0000                .DEFAULT DISPLACEMENT,WORD
                              0000
                              0000      ;+
                              0000      ;         Start up the DR11-W test.  This entails some overhead necessary to cope
                              0000      ;         with both expected and unforseen conditions, figuring out just what
                              0000      ;         devices are to be tested, making sure we can test the indicated devices
                              0000      ;         and setting up writeable space for each device to be tested.
                              0000      ;-
                              0000
                        0000  0000      .ENTRY    UETDR1W00,^M<>                     ; Entry mask
                              0002
         6D   08B9'CF   DE    0002                MOVAL   SSERROR,(FP)               ; Declare exception handler
                              0007                $SETSFM_S ENBFLG = #1              ; Enable system service failure mode
                              0010                $DCLEXH_S DESBLK = EXIT_DESC       ; Declare an exit handler
                              001B
                              001B                $OPEN   FAB = SYSIN_FAB,-          ; Open SYS$INPUT
                              001B                        ERR = RMS_ERROR
                              002A                $CONNECT RAB = SYSIN_RAB,-         ; Connect RAB to SYS$INPUT
                              002A                         ERR = RMS_ERROR
                   02   E1    0039                BBC     S^#DEV$V_TRM,-             ; BR if SYS$INPUT is NOT a terminal
            1E 0418'CF        003B                        SYSIN_FAB+FAB$L_DEV,10$
                              003F                $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
                              003F                          RSLLEN = DEVNAM_LEN,- ; ...a logical name...
                              003F                          RSLBUF = DEVDSC      ; ...for the controller to test
              01   50   D1    0058                CMPL    R0,#SS$_NORMAL             ; Was a controller specified?
                   2E   13    005B                BEQL    PROC_CONT_NAME             ; BR if it was - go process it
                              005D      10$:
                              005D                $GET    RAB = SYSIN_RAB,-          ; Read SYS$INPUT...
                              005D                        ERR = RMS_ERROR            ; ...for the controller name
              044A'CF   B0    006C                MOVW    SYSIN_RAB+RAB$W_RSZ,-      ; Save the name length
              02D2'CF         0070                        DEVNAM_LEN
                   16   12    0073                BNEQ    PROC_CONT_NAME             ; BR if we got something
         02B6'CF   14   D0    0075                MOVL    #SS$_BADPARAM,STATUS       ; Save an exit status if not
              00C4'CF   DF    007A                PUSHAL  NO_CTRLNAME                ; Prepare for message...
                   01   DD    007E                PUSHL   #1                         ; ...
         00741132 8F    DD    0080                PUSHL   #UETP$_TEXT!STS$K_ERROR    ; ...
                   03   DD    0086                PUSHL   #3                         ; ...
                 09B7   31    0088                BRW     ERROR_EXIT                 ; ...to tell of bad setup
                              008B
                              008B      PROC_CONT_NAME:
  0208'CF     02D2'CF   3C    008B                MOVZWL  DEVNAM_LEN,DEVDSC          ; Set the device name length
              0208'CF   DF    0092                PUSHAL  DEVDSC                     ; Make sure...
              0208'CF   DF    0096                PUSHAL  DEVDSC                     ; ...that the specified controller...
   00000000'GF     02   FB    009A                CALLS   #2,G^STR$UPCASE            ; ...is all uppercase for later comparison
  52   0208'CF     01   C1    00A1                ADDL3   #1,DEVDSC,R2               ; Estimate the eventual...
       0210'CF     52   A0    00A7                ADDW2   R2,PROCESS_NAME            ; ...process name length (incl. "_")
                        DE    00AC                MOVAL   PROCESS_NAME+8-            ; Locate first available byte...
                              00AD                        +MAX_PROC_NAME-            ; ...in process name handle...
       50     021C'CF         00AD                        -PROCESS_NAME_FREE,R0      ; ...for device name
                   0B   C3    00B1                SUBL3   #PROCESS_NAME_FREE,-       ; Will the device name fit...
              51   52         00B3                        R2,R1                      ; ...in the remaining space?
                   08   15    00B5                BLEQ    10$                        ; BR if it will
              50   51   C2    00B7                SUBL2   R1,R0                          ; Overwrite handle otherwise...
       0210'CF     0F   B0    00BA                MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
```

```
                                00BF  10$:
          80     5F 8F    90    00BF          MOVB      #^A/ /,(R0)+              ; Separate handle from device name
  60   0227'CF   0208'CF  28    00C3          MOVC3     DEVDSC,DEV_NAME,(R0)      ; Concatenate handle with device name
                      7E  D4    00CB          CLRL      -(SP)                     ; Set the time stamp flag
                 000F'CF  DF    00CD          PUSHAL    TEST_NAME                 ; Set the test name
                      02  DD    00D1          PUSHL     #2                        ; Push the argument count
             00741039 8F  DD    00D3          PUSHL     #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
          00000000'GF 04  FB    00D9          CALLS     #4,G^LIB$SIGNAL           ; Print the startup message
             0002'CF  08  A8    00E0          BISW2     #BEGIN_MSGM,FLAG          ; Set flag so we don't print it again
                                00E5          $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETDR1W00_x
                                00F0
                      02  E1    00F0          BBC       S^#DEV$V_TRM,-            ; BR if SYS$INPUT is NOT a terminal
             66 0418'CF         00F2                    SYSIN_FAB+FAB$L_DEV,20$
                                00F6          $GETDVI_S DEVNAM = SYS$INPUT,-    ; Get the name of...
                                00F6                    EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                                00F6                    ITMLST = INPUT_ITMLST,-
                                00F6                    IOSB   = QUAD_STATUS
             45 02BA'CF   E9    0112          BLBC      QUAD_STATUS,20$           ; Avoid CTRL/C handler if any error
                                0117          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                                0117                    CHAN   = TTCHAN
                                0128          $QIOW_S CHAN       = TTCHAN,-     ; Enable CTRL/C AST's...
                                0128                  FUNC       = #IO$_SETMODE!IO$M_CTRLCAST,-
                                0128                  P1         = CCASTHAND
                 0210'CF  DF    0149          PUSHAL    PROCESS_NAME              ; ...and tell the user...
                      01  DD    014D          PUSHL     #1                        ; ...
             0074832B 8F  DD    014F          PUSHL     #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
          00000000'GF 03  FB    0155          CALLS     #3,G^LIB$SIGNAL           ; ...
                                015C  20$:
                                015C          $TRNLOG_S LOGNAM = MODE,-         ; Get the run mode
                                015C                    RSLLEN = BUFFER_PTR,-
                                015C                    RSLBUF = FAO_BUF
             0014'CF  20  8A    0175          BICB2     #LC_BITM,BUFFER           ; Convert to upper case
       0014'CF   4F 8F    91    017A          CMPB      #^A70/,BUFFER             ; Is this a one shot?
                      05  12    0180          BNEQ      25$                       ; BR if not
             0002'CF  10  A8    0182          BISW2     #ONE_SHOTM,FLAG           ; Set flag for one-shot mode
                                0187  25$:
  0014'CF    504D5544 8F  D1    0187          CMPL      #^A/DUMP/,BUFFER          ; Special dump mode info wanted?
                      05  12    0190          BNEQ      27$                       ; BR if not
             0002'CF  20  A8    0192          BISW2     #DUMP_MODEM,FLAG          ; Set flag for dump mode messages
                                0197  27$:
                                        0197  ;
                                        0197  ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                        0197  ; and unit configuration and lets us know if the setup to run this test was
                                        0197  ; done correctly.
                                        0197  ;
                                        0197              $OPEN    FAB = INI_FAB,-            ; Open file 'UETINIDEV.DAT''
                                        0197                       ERR = RMS_ERROR
                                        01A6              $CONNECT RAB = INI_RAB,-            ; Connect the RAB and FAB
                                        01A6                       ERR = RMS_ERROR
                                        01B5              $MGBLSC_S  INADR = INADDRESS,-      ; Connect to UETSUPDEV global section
                                        01B5                        RETADR = OUTADDRESS,-
                                        01B5                        GSDNAM = SUPDEV_GBLSEC,-
                                        01B5                         FLAGS = #SEC$M_EXPREG
                 00000978 8F   50   D1  01D4              CMPL     R0,#SS$_NOSUCHSEC          ; Was the section already there?
                               37   12  01DB              BNEQ     30$                        ; BR if it was...
                                        01DD              $OPEN    FAB = SUP_FAB,-            ; ...else open ''UETSUPDEV.DAT''
                                        01DD                       ERR = RMS_ERROR
                                        01EC              $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                                        01EC                       INADR = INADDRESS,-
                                        01EC                      RETADR = OUTADDRESS,-
                                        01EC                      GSDNAM = SUPDEV_GBLSEC,-
                                        01EC                       FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                                        0214  30$:
           56   02CE'CF   02CA'CF   C3  0214              SUBL3    OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                                        021C
                                        021C  FIND_IT:
                                        021C              $GET     RAB = INI_RAB,-            ; Get the first record
                                        021C                       ERR = RMS_ERROR
                          01ED'CF   DF  022B              PUSHAL   CONT_DESC                  ; Make sure...
                          01ED'CF   DF  022F              PUSHAL   CONT_DESC                  ; ...that the controller name...
                 00000000'GF   02   FB  0233              CALLS    #2,G^STR$UPCASE            ; ...is all uppercase letters
                   0014'CF  44 8F   91  023A              CMPB     #^A/D/,BUFFER              ; Is this a DDB?
                               27   13  0240              BEQL     10$                        ; Go on if not
                   0014'CF  45 8F   91  0242              CMPB     #^A/E/,BUFFER              ; Is this the end of the file?
                               D2   12  0248              BNEQ     FIND_IT                    ; Continue on if not
                          0208'CF   DF  024A              PUSHAL   DEVDSC                     ; Push device not supported message
                          0210'CF   DF  024E              PUSHAL   PROCESS_NAME               ; Parameters on the stack
                               02   DD  0252              PUSHL    #2
                      00748333 8F   DD  0254              PUSHL    #UETP$_DENOSU
                               02   F0  025A              INSV     #STS$K_ERROR,-             ; Set the severity code...
                               00       025C                       #STS$V_SEVERITY,-
                          6E   03       025D                       #STS$S_SEVERITY,(SP)
                      02B6'CF  6E   D0  025F              MOVL     (SP),STATUS                ; ...and save it as the exit status
                               04   DD  0264              PUSHL    #4
                             07D9   31  0266              BRW      ERROR_EXIT                 ; Exit in error
                                        0269  10$:
      0227'CF   001A'CF   02D2'CF   29  0269              CMPC     DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
                               A7   12  0273              BNEQ     FIND_IT                    ; BR if not
           0500'CF   04CC'CF   06   28  0275              MOVC3    #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the record file address
                   0018'CF  54 8F   91  027D              CMPB     #^A/T/,BUFFER+4            ; Can we test this controller?
                               2F   13  0283              BEQL     FOUND_IT                   ; BR if we can...
                                        0285              $FAO_S   CTRSTR = DEAD_CTRLNAME,-   ; ...and yell at user if we can't
                                        0285                       OUTLEN = BUFFER_PTR,-
                                        0285                       OUTBUF = FAO_BUF,-
                                        0285                       P1     = #DEVDSC
                      02B6'CF  14   D0  029E              MOVL     #SS$_BADPARAM,STATUS       ; Set return status
                          000C'CF   DF  02A3              PUSHAL   BUFFER_PTR                 ; ...
```

```
                  01     DD  02A7                PUSHL    #1                          ; ...
         00741132 8F     DD  02A9                PUSHL    #UETP$_TEXT!STS$K_ERROR     ; ...
                  03     DD  02AF                PUSHL    #3                          ; ...
                078E     31  02B1                BRW      ERROR_EXIT                  ; We can't test what we can't test
                             02B4
                             02B4       FOUND_IT:
                             02B4                $GET     RAB = INI_RAB,-             ; Get a record
                             02B4                         ERR = RMS_ERROR
             01ED'CF     DF  02C3                PUSHAL   CONT_DESC                   ; Make sure...
             01ED'CF     DF  02C7                PUSHAL   CONT_DESC                   ; ...that this line...
       00000000'GF  02   FB  02CB                CALLS    #2,G^STR$UPCASE             ; ...is all uppercase letters
          0014'CF   55 8F    91  02D2            CMPB     #^A/U/,BUFFER               ; Is this a UCB?
                  24     13  02D8                BEQL     30$                         ; BR if it is
          0014'CF   44 8F    91  02DA            CMPB     #^A/D/,BUFFER               ; Is this a DDB?
                  19     13  02E0                BEQL     20$                         ; BR if yes
          0014'CF   45 8F    91  02E2            CMPB     #^A/E/,BUFFER               ; Is this the end?
                  11     13  02E8                BEQL     20$                         ; BR if yes
                             02EA       10$:
             0151'CF     DF  02EA                PUSHAL   ILLEGAL_REC                 ; Then this is an error in the record
                  01     DD  02EE                PUSHL    #1                          ; Push the error message
         00741132 8F     DD  02F0                PUSHL    #UETP$_TEXT!STS$K_ERROR     ; Push the signal name
                  03     DD  02F6                PUSHL    #3                          ; Push the temp arg count
                0747     31  02F8                BRW      ERROR_EXIT                  ; Finish for good
                             02FB       20$:
                015C     31  02FB                BRW      ALL_SET                     ; Found DDB or END
                             02FE       30$:
          0018'CF   54 8F    91  02FE            CMPB     #^A/T/,BUFFER+4             ; Is the unit testable?
                  AE     12  0304                BNEQ     FOUND_IT                    ; BR if not
            05    20     3B  0306                SKPC     #^A/ 7,#MAX_UNIT_DESIG,-    ; Find out where unit number really is
             001A'CF         0309                         BUFFER+6
                  50     D7  030C                DECL     R0                          ; Units must all be at least one digit
       61    50   30     3B  030E                SKPC     #^A/0/,R0,(R1)              ; Skip leading zeroes on the unit
                  50     D6  0312                INCL     R0                          ; Compensate for DECL above
 0208'CF   02D2'CF  50   A1  0314                ADDW3    R0,DEVNAM_LEN,DEVDSC        ; Calculate de;ice'unit string length
           52  02D2'CF   3C  031C                MOVZWL   DEVNAM_LEN,R2               ; Offset to unit number in DEVDSC
   0227'C2   61   50     28  0321                MOVC3    R0,(R1),DEV_NAME(R2)        ; Append unit number to device
                             0327                $GETDEV_S DEVNAM = DEVDSC,-          ; Get the device characteristics
                             0327                            PRIBUF = DIB
       57   0242'CF      9A  033C                MOVZBL   DIBBUF+DIB$B_DEVCLASS,R7    ; Save the device class
       58   0243'CF      9A  0341                MOVZBL   DIBBUF+DIB$B_DEVTYPE,R8     ; Save the device type
                             0346                $FAO_S   CTRSTR = CS1,-
                             0346                         OUTBUF = FAO_BUF,-
                             0346                         P1     = R7,-
                             0346                         P2     = R8                 ; Make it into a string
 02CA'DF   56   0014'CF   06   39  035B          MATCHC   #6,BUFFER,R6,@OUTADDRESS    ; Find the device class and type
                          1E   13  0364          BEQL     40$                         ; BR if it was found
                             0366                $FAO_S   CTRSTR = CS3,-              ; Try for full class support
                             0366                         OUTBUF = FAO_BUF,-
                             0366                            P1 = R7
 02CA'DF   56   0014'CF   06   39  0379          MATCHC   #6,BUFFER,R6,@OUTADDRESS    ; Find the device class only
                          0D   12  0382          BNEQ     50$                         ; BR if not found
                             0384       40$:
       55   000F'CF      9A  0384                MOVZBL   TEST_NAME,R5                ; Get the test name length
   0017'CF   63   55     29  0389                CMPC3    R5,(R3),TEST_NAME+8         ; Are we the right test?
                  1F     13  038F                BEQL     60$                         ; BR if yes
                             0391       50$:
             0208'CF     DF  0391                PUSHAL   DEVDSC                      ; Push device not supported message
```

```
        0210'CF   DF  0395         PUSHAL  PROCESS_NAME              ; Parameters on the stack
             02   DD  0399         PUSHL   #2                        ; Push the argument count
    00748333 8F   DD  039B         PUSHL   #UETP$_DENOSU
             02   F0  03A1         INSV    #STS$K_ERROR,-
             00       03A3                 #STS$V_SEVERITY,-
          6E 03       03A4                 #STS$S_SEVERITY,(SP)      ; Set the severity code...
   02B6'CF   6E   D0  03A6         MOVL    (SP),STATUS               ; ...and save it as the exit status
             04   DD  03AB         PUSHL   #4                        ; Push the partial arg count...
           0692   31  03AD         BRW     ERROR_EXIT                ; ...and split this scene
                      03B0    60$:
                      03B0         $EXPREG_S PAGCNT = #PAGES,-       ; Get a new node of demand zero memory
                      03B0               RETADR = NEW_NODE
0300'CF   0308'DF  5D 03C1         INSQTI  @NEW_NODE,UNIT_LIST       ; Put the new node in the unit list
     56   0308'CF  D0 03C8         MOVL    NEW_NODE,R6               ; Save a copy of its address
       08 A6   01  90 03CD         MOVB    #1,UETUNT$B_TYPE(R6)      ; Set the structure type
        09F1   8F  B0 03D1         MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
           09 A6      03D5                 UETUNT$W_SIZE(R6)         ; Set the structure size
  14 A6   0208'CF  90 03D7         MOVB    DEVDSC,UETUNT$T_FILSPC(R6) ; Set the device name size
020C'DF   0208'CF  28 03DD         MOVC3   DEVDSC,@DEVDSC+4,-        ; Save the device name
             15 A6    03E4                 UETUNT$T_FILSPC+1(R6)
                02 88 03E6         BISB2   #UETUNT$M_TESTABLE,-      ; Assume DR11-W testable
             0B A6    03E8                 UETUNT$B_FLAGS(R6)
                      03EA         $ASSIGN_S DEVNAM = DEVDSC,-       ; Get the DR11-W for our exclusive use
                      03EA               CHAN   = UETUNT$W_CHAN(R6)
          3E 50   E8  03FA         BLBS    R0,70$                    ; We're OK if we got the device
   02B6'CF   50   D0  03FD         MOVL    R0,STATUS                 ; Save the failure code as exit status
             02   F0  0402         INSV    #STS$K_ERROR,-
             00       0404                 #STS$V_SEVERITY,-
   02B6'CF   03       0405                 #STS$S_SEVERITY,STATUS    ; Set the severity code
             02   8A  0409         BICB2   #UETUNT$M_TESTABLE,-      ; We can't test DR11-W
          0B A6       040B                 UETUNT$B_FLAGS(R6)
                      040D         $FAO_S  CTRSTR = DEVALLOC,-       ; Otherwise bitch somewhat
                      040D                 OUTLEN = BUFFER_PTR,-
                      040D                 OUTBUF = FAO_BUF,-
                      040D                 P1     = #DEVDSC
        000C'CF   DF  0426         PUSHAL  BUFFER_PTR                ; ...
             01   DD  042A         PUSHL   #1                        ; ...
    00741132 8F   DD  042C         PUSHL   #UETP$_TEXT!STS$K_ERROR   ; ...
        02B6'CF   DD  0432         PUSHL   STATUS                    ; ...
             04   DD  0436         PUSHL   #4
           0607   31  0438         BRW     ERROR_EXIT
                      043B    70$:
             34   28  043B         MOVC3   #4*<QIO$_NARGS+1>,-       ; Fill static $QIO_G args...
01BC C6   0529'CF     043D                 DUMMY_QIO,XA_K_QIO(R6)
          0C A6   3C  0443         MOVZWL  UETUNT$W_CHAN(R6),-       ; ...with those which...
        01C4 C6       0446                 XA_K_QIO+QIO$_CHAN(R6)
        01A4 C6   7E  0449         MOVAQ   XA_Q_IOSB(R6),-           ; ...can't be filled at assembly
        01CC C6       044D                 XA_K_QIO+QIO$_IOSB(R6)
        0242'CF   7D  0450         MOVQ    DIBBUF+DIB$B_DEVCLASS,-   ; Save original characs
        01B4 C6       0454                 XA_Q_ORIGINAL(R6)
           FE5A   31  0457         BRW     FOUND_IT                  ; Do the next UCB
```

```
                          045A    ;
                          045A    ; Arrive here when we have the device configuration.  In normal or loop forever
                          045A    ; mode, set a timer far enough in the future such that we can do a reasonable
                          045A    ; set of tests before the timer expires, but if our device gets hung, the
                          045A    ; program won't waste too much time before noticing.  Let one-shot mode be a
                          045A    ; special case.
                          045A    ;
                          045A    ALL_SET:
          0300'CF     D5  045A            TSTL    UNIT_LIST                   ; Anything to test?
               16     12  045E            BNEQ    10$                         ; BR if yes
          012B'CF     DF  0460            PUSHAL  NOUNIT_SELECTED             ; Else set up the error message...
               01     DD  0464            PUSHL   #1                          ; ...argument count...
      00741132 8F     DD  0466            PUSHL   #UETP$_TEXT!STS$K_ERROR     ; ...signal name...
               03     DD  046C            PUSHL   #3                          ; ...and parameter count
     02B6'CF   14     D0  046E            MOVL    #SS$_BADPARAM,STATUS        ; Set return status
             05CC     31  0473            BRW     ERROR_EXIT                  ; ...and give up, complaining
                          0476    10$:
     0002'CF   04     A8  0476            BISW2   #SAFE_TO_UPDM,FLAG          ; OK, safe to update UETINIDEV.DAT now
                          047B    ;       MOVL    DEVNAM_LEN,DEVDSC           ; DEVDSC will describe device name
  05 0002'CF   04     E1  047B            BBC     #ONE_SHOTV,FLAG,TIME_IT     ; BR if in normal loop forever modes
     0002'CF   02     A8  0481            BISW2   #TEST_OVERM,FLAG            ; One-shot mode, stop after one shot!
                          0486    ; Because not all $QIOs have a timeout parameter, this test will always fall
                          0486    ; into TIME_IT to do a $SETIMR.
                          0486
                          0486    TIME_IT:
                          0486            $SETIMR_S DAYTIM = THREEMIN,-       ; Set timer AST to 3 minutes
                          0486                    ASTADR = TIME_OUT,-
                          0486                    EFN    = #EFN2
```

```
                                0499          .SBTTL  Test the DR11-W
                                0499  RESTART:
                                0499  ;
                                0499  ; At this point the device designation is in location DEV_NAME pointed to by
                                0499  ; descriptor DEVDSC.  The device is known to be supported by this test.
                                0499  ;
         56   0300'CF      DE   0499          MOVAL   UNIT_LIST,R6              ; R6 will point to the current node
                                049E  TEST_LOOP:
              56   66      C0   049E          ADDL2   (R6),R6                   ; Point to the next possible node
     56   00000300'8F      D1   04A1          CMPL    #UNIT_LIST,R6             ; Back at the head of the queue?
                   03      12   04A8          BNEQ    10$                       ; BR if not
                 01DE      31   04AA          BRW     90$                       ; Exit test portion if we are
                                04AD  10$:
                   01      E1   04AD          BBC     #UETUNT$V_TESTABLE,-      ; Skip this unit if can't test it
             EC 0B A6           04AF                  UETUNT$B_FLAGS(R6),TEST_LOOP
                                04B2
     58   000000FA 8F      D0   04B2          MOVL    #<WRITE_SIZE/4>,R8        ; Fill alternate words (byte count/4)...
         59   09F2 C6      DE   04B9          MOVAL   XA_K_BUF(R6),R9           ; ...of the read/write buffer...
                                04BE  20$:
  02D4'CF    02D8'CF       C0   04BE          ADDL2   RANDOM2,RANDOM1           ; ...with random...
       89    02D4'CF       3C   04C5          MOVZWL  RANDOM1,(R9)+             ; ...words...
                F1 58      F5   04CA          SOBGTR  R8,20$                    ; ...until it's full
                                04CD  ;
                                04CD  ; Set up DR11-W quadword characteristics buffer for future IO$_SETMODEs.
                                04CD  ; Copy device type, class and (bogus) buffer size and enabling for Unibus BDP
                                04CD  ; when we want it.
                                04CD  ;
              01B4 C6      7D   04CD          MOVQ    XA_Q_ORIGINAL(R6),-       ; Class, type & transfer count
              01AC C6           04D1                  XA_Q_CHARAC(R6)
                   02      88   04D4          BISB2   #XA$M_LINK,-              ; Add other characteristics...
              01B0 C6           04D6                  XA_Q_CHARAC+4(R6)         ; ...(but avoid XA$M_DATAPATH)
                                04D9  ;
                                04D9  ; As described previously, the TABLE_GEN and LINE_GEN macros set up a set of
                                04D9  ; parallel tables with parameters to be used in $QIOs to the DR11-W.  Go
                                04D9  ; through those tables.  For each $QIO which transfers data, check that the
                                04D9  ; data were passed correctly and clear the words into which data were written.
                                04D9  ;
                                04D9  ; Although the data structures of this test would permit multi-unit,
                                04D9  ; asynchronous testing of DR11-W's, the way VMS treats DR11-W's (one unit per
                                04D9  ; logical controller) means that we will test one one unit per test invocation.
                                04D9  ; Since operations will turn out synchronous anyway, we may as well be honest
                                04D9  ; and use the $QIOW system service to synchronize control.
                                04D9  ;
                   57      D4   04D9          CLRL    R7                        ; Set up counter to go thru tables
         5A   01BC C6      DE   04DB          MOVAL   XA_K_QIO(R6),R10          ; Point to $QIO arglist for clarity
                                04E0  30$:
    2C AA   0608'CF47      D0   04E0          MOVL    P5_TABLE[R7],QIO$_P5(R10) ; Set up those $QIO_G args...
    28 AA   05EC'CF47      D0   04E7          MOVL    P4_TABLE[R7],QIO$_P4(R10) ; ...which must be done for...
    24 AA   05D0'CF47      D0   04EE          MOVL    P3_TABLE[R7],QIO$_P3(R10) ; ...each individual $QIO
    20 AA   05B4'CF47      D0   04F5          MOVL    P2_TABLE[R7],QIO$_P2(R10) ; ...
 1C AA   0598'CF47   56    C1   04FC          ADDL3   R6,P1_TABLE[R7],QIO$_P1(R10) ; ...
             18 AA   57    D0   0504          MOVL    R7,QIO$_ASTPRM(R10)       ; ...
    14 AA   057C'CF47      D0   0508          MOVL    ASTADR_TABLE[R7],QIO$_ASTADR(R10) ; ...
    0C AA   0560'CF47      D0   050F          MOVL    FUNC_TABLE[R7],QIO$_FUNC(R10) ; ...
                                0516          $QIOW_G (R10)                     ; Do one function of the DR11-W
                10 A6      D6   051D          INCL    UETUNT$L_ITER(R6)         ; Count $QIOs done
```

```
          51   01F0 C6   9A  0520    918         MOVZBL  XA_B_TSI(R6),R1                  ; Put index value into index register
   01F1 C641   0310'CF   7D  0525    919         MOVQ    DIAG_BUF,XA_K_TSTAMP(R6)[R1] ; Store $QIO starting time stamp
                    51   D6  052D    920         INCL    R1                               ; Bump up index
   01F1 C641   0318'CF   7D  052F    921         MOVQ    DIAG_BUF+8,XA_K_TSTAMP(R6)[R1] ; Store $QIO ending time stamp
                    51   D6  0537    922         INCL    R1                               ; Bump up index
        01F0 C6     51   90  0539    923         MOVB    R1,XA_B_TSI(R6)                  ; Keep index modulus 2**8
                             053E    924
                 38 50   E8  053E    925         BLBS    R0,40$                           ; BR if the $QIO itself worked
         0788'CF    00   FB  0541    926         CALLS   #0,DEBUG_DUMP                    ; Print diagnostic info if it failed
         02B6'CF    50   D0  0546    927         MOVL    R0,STATUS                        ; Save a record of what failed...
            58    14 A6  DE  054B    928         MOVAL   UETUNT$T_FILSPC(R6),R8           ; ...
                             054F    929         $FAO_S  CTRSTR = QIO_ERROR,-             ; ... and consider it a fatal error
                             054F    930                 OUTLEN = BUFFER_PTR,-
                             054F    931                 OUTBUF = FAO_BUF,-
                             054F    932                 P1     = R8
               02B6'CF   DD  0564    933         PUSHL   STATUS                           ; ...
               000C'CF   DF  0568    934         PUSHAL  BUFFER_PTR                       ; ...
                    01   DD  056C    935         PUSHL   #1                               ; ...
         00741132 8F     DD  056E    936         PUSHL   #UETP$_TEXT!STS$K_ERROR          ; ...
                    04   DD  0574    937         PUSHL   #4                               ; ...
                  04C9   31  0576    938         BRW     ERROR_EXIT                       ; ...
                             0579    939 40$:
       71 01A4 C6        E8  0579    940         BLBS    XA_Q_IOSB(R6),50$                ; BR if the function of the $QIO worked
         0788'CF    00   FB  057E    941         CALLS   #0,DEBUG_DUMP                    ; Type special info if in DUMP mode
            58    14 A6  DE  0583    942         MOVAL   UETUNT$T_FILSPC(R6),R8
                             0587    943         $FAO_S  CTRSTR = QIO_FUNC_FAIL,- ; Report the problem otherwise
                             0587    944                 OUTLEN = BUFFER_PTR,-
                             0587    945                 OUTBUF = FAO_BUF,-
                             0587    946                 P1     = R8,-
                             0587    947                 P2     = FUNC_TABLE[R7],-
                             0587    948                 P3     = XA_Q_IOSB(R6),-
                             0587    949                 P4     = XA_Q_IOSB+4(R6)
        7E     01A4 C6   3C  05A9    950         MOVZWL  XA_Q_IOSB(R6),-(SP)              ; ...
               000C'CF   DF  05AE    951         PUSHAL  BUFFER_PTR                       ; ...
         000F0001 8F     DD  05B2    952         PUSHL   #^XF0001                         ; ...
         00741130 8F     DD  05B8    953         PUSHL   #UETP$_TEXT                      ; ...
               01A4 C6   F0  05BE    954         INSV    XA_Q_IOSB(R6),-                  ; Set the severity code
                    00       05C2    955                 #STS$V_SEVERITY,-
              6E    03       05C3    956                 #STS$S_SEVERITY,(SP)
               02B2'CF   D6  05C5    957         INCL    ERROR_COUNT                      ; Keep a running count...
               02B2'CF   DD  05C9    958         PUSHL   ERROR_COUNT                      ; ...of the errors we've gotten
               0210'CF   DF  05CD    959         PUSHAL  PROCESS_NAME
         00010002 8F     DD  05D1    960         PUSHL   #^X10002
         00748022 8F     DD  05D7    961         PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; Have the error stand out
     00000000'GF    08   FB  05DD    962         CALLS   #8,G^LIB$SIGNAL                  ; Bitch, bitch, bitch
                    02   8A  05E4    963         BICB2   #UETUNT$M_TESTABLE,-             ; Indicate that this unit is no good
                 0B A6       05E6    964                 UETUNT$B_FLAGS(R6)
   0002'CF     0040 8F   A8  05E8    965         BISW2   #NO_MESSAGEM,FLAG                ; Indicate message already printed
                             05EF    966 50$:
```

```
                05B4'CF47   D5  05EF   968        TSTL    P2_TABLE[R7]               ; If zero length data transfer (words)...
                       7F   13  05F4   969        BEQL    80$                        ; ...skip the check and reset
   58   05B4'CF47   FE 8F   78  05F6   970        ASHL    #-2,P2_TABLE[R7],R8        ; Convert byte count to every-other-word cou
      59   0598'CF47   56   C1  05FE   971        ADDL3   R6,P1_TABLE[R7],R9         ; Check that the data buffer...
                                0605   972 60$:
             5B   69   89   AD  0605   973        XORW3   (R9)+,(R9),R11             ; ...got filled correctly
                       65   13  0609   974        BEQL    70$                        ; BR if it did
          5F  0002'CF  06   E0  060B   975        BBS     #NO_MESSAGEV,FLAG,70$      ; Avoid extra messages
              0788'CF  00   FB  0611   976        CALLS   #0,DEBUG_DUMP              ; Type special info if in DUMP mode
      5B     5B   10   00   EA  0616   977        FFS     #0,#16,R11,R11             ; Find the first bit of bad data
        5B     5B   FD 8F   78  061B   978        ASHL    #-3,R11,R11                ; Convert bit-position to byte-in-word
                7E   694B   9A  0620   979        MOVZBL  (R9)[R11],-(SP)            ; Save the bad byte,...
             7E   FE A94B   9A  0624   980        MOVZBL  -2(R9)[R11],-(SP)          ; ...the corresponding good byte,...
             7E   59   56   C3  0629   981        SUBL3   R6,R9,-(SP)                ; ...where the bad data was...
                  6E   5B   C0  062D   982        ADDL2   R11,(SP)                   ; ...
         6E    0598'CF47    C2  0630   983        SUBL2   P1_TABLE[R7],(SP)          ; ...in our buffer,...
                  0208'CF   DF  0636   984        PUSHAL  DEVDSC                     ; ...the device name,...
             000F0004 8F    DD  063A   985        PUSHL   #^XF0004                   ; ...the argument count,...
             0074801A 8F    DD  0640   986        PUSHL   #UETP$_DATADEVERR!STS$K_ERROR ; ...and the error type
                  02B2'CF   D6  0646   987        INCL    ERROR_COUNT                ; Keep a running count...
                  02B2'CF   DD  064A   988        PUSHL   ERROR_COUNT                ; ...of the errors we've gotten
                  0210'CF   DF  064E   989        PUSHAL  PROCESS_NAME
             00010002 8F    DD  0652   990        PUSHL   #^X10002
             00748022 8F    DD  0658   991        PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; Have the error stand out
         00000000'GF   0A   FB  065E   992        CALLS   #10,G^LIB$SIGNAL           ; Bitch, bitch, bitch
                       02   8A  0665   993        BICB2   #UETUNT$M_TESTABLE,-       ; Indicate that this unit is no good
                    0B A6       0667   994                UETUNT$B_FLAGS(R6)
        0002'CF   0040 8F   A8  0669   995        BISW2   #NO_MESSAGEM,FLAG          ; Indicate message already printed
                                0670   996 70$:
                       89   B4  0670   997        CLRW    (R9)+                      ; Clear word to which DR11-W does DATO
                    90 58   F5  0672   998        SOBGTR  R8,60$                     ; Loop through the whole buffer
                                0675   999 80$:
        0002'CF   0040 8F   AA  0675  1000        BICW2   #NO_MESSAGEM,FLAG          ; New buffer rates another message
          09 0002'CF   01   E0  067C  1001        BBS     #TEST_OVERV,FLAG,90$       ; Exit quickly if the test is over
        FE58 57   01   06   9D  0682  1002        ACBB    #LINE_GEN_COUNT-1,#1,R7,30$ ; Loop until all $QIOs are done
                                0688  1003
                     FE13   31  0688  1004        BRW     TEST_LOOP                  ; Loop for next DR11-W
                                068B  1005
                                0683  1006 90$:
                     01CE   30  068B  1007        BSBW    RESET_DR11WS               ; Do $QIOs to reset original characs
                02DC'CF  D6 068E             INCL    ITERATION                     ; Increment iteration count
  56   0300'CF  00000300'8F  C1 0692             ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Go through UNIT_LIST to...
                            069C   100$:
                     01  E0 069C             BBS     #UETUNT$V_TESTABLE,-          ; ...see if any testable units are left
               11 0B A6     069E                     UETUNT$B_FLAGS(R6),110$ ; ...and BR if at least one is
                  56 66  C0 06A1             ADDL2   (R6),R6                       ; This one isn't.  Is there another?
        56  00000300'8F  D1 06A4             CMPL    #UNIT_LIST,R6
                     EF  12 06AB             BNEQ    100$                          ; BR if there are others to try
            0002'CF  02  A8 06AD             BISW2   #TEST_OVERM,FLAG              ; None testable so indicate test over
                            06B2   110$:
            0002'CF  02  B3 06B2             BITW    #TEST_OVERM,FLAG              ; Is the test over?
                     03  12 06B7             BNEQ    SUC_EXIT                      ; BR if yes
                   FDDD  31 06B9             BRW     RESTART                       ; Loop until the test is over
                            06BC
                            06BC   SUC_EXIT:
         4C 0002'CF  04  E0 06BC             BBS     #ONE_SHOTV,FLAG,30$           ; Skip minimum I/O check if one-shot
            56   0300'CF DE 06C2             MOVAL   UNIT_LIST,R6                  ; Check the queue...
                            06C7   20$:
                  56 66  C0 06C7             ADDL2   (R6),R6                       ; ...to see if...
        56  00000300'8F  D1 06CA             CMPL    #UNIT_LIST,R6                 ; ...each DR11-W...
                     3B  13 06D1             BEQL    30$                           ; ...tested...
     10 A6  00000258 8F  D1 06D3             CMPL    #MINIMUM,UETUNT$L_ITER(R6) ; ...has done the minimum I/O...
                     EA  15 06DB             BLEQ    20$                           ; ...to not be considered hung
               57  14 A6 DE 06DD             MOVAL   UETUNT$T_FILSPC(R6),R7
                            06E1             $FAO_S  CTRSTR = SLOW_DR11W,-        ; Otherwise bitch somewhat
                            06E1                     OUTLEN = BUFFER_PTR,-
                            06E1                     OUTBUF = FAO_BUF,-
                            06E1                     P1     = R7,=
                            06E1                     P2     = UETUNT$L_ITER(R6)
                000C'CF  DF 06F9             PUSHAL  BUFFER_PTR                    ; ...
                     01  DD 06FD             PUSHL   #1                            ; ...
            00741130 8F  DD 06FF             PUSHL   #UETP$_TEXT!STS$K_WARNING ; ...
       00000000'GF   03  FB 0705             CALLS   #3,G^LIB$SIGNAL               ; ...
                     B9  11 070C             BRB     20$
                            070E   30$:
                            070E             $TRNLOG_S LOGNAM = MODE,-
                            070E                       RSLLEN = BUFFER_PTR,-
                            070E                       RSLBUF = FAO_BUF       ; Get the run mode
            0014'CF  20  8A 0727             BICB2   #LC_BITM,BUFFER               ; Convert to upper case
       0014'CF    4C 8F  91 072C             CMPB    #^A7L/,BUFFER                 ; Is this a loop for ever?
                     40  12 0732             BNEQ    10$                           ; BR if not
            0002'CF  02  AA 0734             BICW2   #TEST_OVERM,FLAG              ; Reset the termination flag
                02E0'CF  D6 0739             INCL    PASS                          ; Bump the pass count
                            073D             $FAO_S  CTRSTR = PASS_MSG,-
                            073D                     OUTLEN = BUFFER_PTR,-
                            073D                     OUTBUF = FAO_BUF,-
                            073D                     P1     = PASS,-
                            073D                     P2     = ITERATION,-
                            073D                     P3     = #0                  ; Make the end of pass message
                000C'CF  DF 075A             PUSHAL  BUFFER_PTR                    ; Push the string desc.
                     01  DD 075E             PUSHL   #1                            ; Push arg count
            00741133 8F  DD 0760             PUSHL   #UETP$_TEXT!STS$K_INFO        ; Push the signal name
       00000000'GF   03  FB 0766             CALLS   #3,G^LIB$SIGNAL               ; Print the end of pass message
                02DC'CF  D4 076D             CLRL    ITERATION                     ; Reset the iteration count
                   FD12  31 0771             BRW     TIME_IT                       ; Do the next pass
                            0774   10$:
  02B6'CF   10000001 8F  D0 0774             MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
```

```
                    077D  1066          $EXIT_S STATUS                          ; Exit with the status
```

```
                             0788           .SBTTL  Routine to Dump Debugging Info
                             0788      ;++
                             0788      ; FUNCTIONAL DESCRIPTION:
                             0788      ;        This routine will be called only if the logical name MODE translates to
                             0788      ;        the string "DUMP", and then only for those situations where a dump of
                             0788      ;        info pertaining to the last $QIO would be useful.
                             0788      ;
                             0788      ; CALLING SEQUENCE:
                             0788      ;        CALLS   #0,DEBUG_DUMP
                             0788      ;
                             0788      ; INPUT PARAMETERS:
                             0788      ;        NONE
                             0788      ;
                             0788      ; IMPLICIT INPUTS:
                             0788      ;        R0 has the $QIO (in)completion status.
                             0788      ;        R7 is an index into the $QIO tables.
                             0788      ;        R10 points to the $QIO argument list.  (redundant with R6)
                             0788      ;        XA_Q_IOSB(R6) is the IOSB for the last $QIO.
                             0788      ;
                             0788      ; OUTPUT PARAMETERS:
                             0788      ;        NONE
                             0788      ;
                             0788      ; IMPLICIT OUTPUTS:
                             0788      ;        Message to SYS$OUTPUT
                             0788      ;        R0 and R1 returned unscathed!
                             0788      ;
                             0788      ; COMPLETION CODES:
                             0788      ;        NONE
                             0788      ;
                             0788      ; SIDE EFFECTS:
                             0788      ;        BUFFER and BUFFER_PTR modified after $FAO call
                             0788      ;
                             0788      ;--
                             0788
                             0788      DEBUG_DUMP:
                     0FFC    0788           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                             078A
01 0002'CF     05      E0    078A           BBS     #DUMP_MODEV,FLAG,10$    ; Dump mode info wanted?
                       04    0790           RET                             ; Nope - call is a no-op
                             0791      10$:
               03      BB    0791           PUSHR   #^M<R0,R1>              ; Save all registers!
                             0793           $FAO_S  CTRSTR = DEBUG_MSG,-    ; All the news that fits, we print
                             0793                   OUTLEN = BUFFER_PTR,-
                             0793                   OUTBUF = FAO_BUF,-
                             0793                   P1     = R0,-
                             0793                   P2     = QIO$_EFN(R10),-
                             0793                   P3     = QIO$_CHAN(R10),-
                             0793                   P4     = QIO$_FUNC(R10),-
                             0793                   P5     = QIO$_ASTADR(R10),-
                             0793                   P6     = QIO$_ASTPRM(R10),-
                             0793                   P7     = QIO$_P1(R10),-
                             0793                   P8     = QIO$_P2(R10),-
                             0793                   P9     = QIO$_P3(R10),-
                             0793                   P10    = QIO$_P4(R10),-
                             0793                   P11    = QIO$_P5(R10),-
                             0793                   P12    = XA_Q_IOSB(R6),-
                             0793                   P13    = XA_Q_IOSB+4(R6),-
                             0793  1125           P14     = XA_Q_CHARAC(R6),-
                             0793  1126           P15     = XA_Q_CHARAC+4(R6)
           000C'CF       DF  07D6  1127           PUSHAL  BUFFER_PTR
        00010001 8F      DD  07DA  1128           PUSHL   #^X10001
        00741133 8F      DD  07E0  1129           PUSHL   #UETP$_TEXT!STS$K_INFO
     00000000'GF 03      FB  07E6  1130           CALLS   #3,G^LIB$SIGNAL
        51   01F0 C6     9A  07ED  1131           MOVZBL  XA_B_TSI(R6),R1          ; Index of next time stamp quad pair
             51   02     C2  07F2  1132           SUBL2   #2,R1                    ; Step back to current time stamp quad pair
   52    0000004C 8F     D0  07F5  1133           MOVL    #<19*4>,R2               ; First available longword in DIAG_BUF
             53   02     D0  07FC  1134           MOVL    #2,R3                    ; Number of quad pairs for DIAG_MSG
                             07FF  1135 20$:
             51   02     C2  07FF  1136           SUBL2   #2,R1                    ; Step back one quad pair
                  07     18  0802  1137           BGEQ    30$                      ; BR if not below first entry
   51    000000FE 8F     D0  0804  1138           MOVL    #TIME_STAMP_LEN-2,R1     ; Point to top of quad pair list if it is
                             080B  1139 30$:
0310'C2   01F1 C641      7D  080B  1140           MOVQ    XA_K_TSTAMP(R6)[R1],DIAG_BUF(R2) ; Position one time stamp for $FAO
             52   08     C0  0813  1141           ADDL2   #8,R2                    ; Next available longword in DIAG_BUF
                  51     D6  0816  1142           INCL    R1                       ; Bump index
0310'C2   01F1 C641      7D  0818  1143           MOVQ    XA_K_TSTAMP(R6)[R1],DIAG_BUF(R2) ; Position one time stamp for $FAO
             52   08     C0  0820  1144           ADDL2   #8,R2                    ; Next available longword in DIAG_BUF
                  51     D7  0823  1145           DECL    R1                       ; Bump index back where expected
               D7 53     F5  0825  1146           SOBGTR  R3,20$
                             0828  1147           $FAOL_S CTRSTR = DIAG_MSG,-
                             0828  1148                   OUTLEN = BUFFER_PTR,-
                             0828  1149                   OUTBUF = FAO_BUF,-
                             0828  1150                   PRMLST = DIAG_BUF
           000C'CF       DF  083F  1151           PUSHAL  BUFFER_PTR
        00010001 8F      DD  0843  1152           PUSHL   #^X10001
        00741133 8F      DD  0849  1153           PUSHL   #UETP$_TEXT!STS$K_INFO
     00000000'GF 03      FB  084F  1154           CALLS   #3,G^LIB$SIGNAL
                  03     BA  0856  1155           POPR    #^M<R0,R1>               ; Restore registers pristine
                         04  0858  1156           RET
```

```
          0859           .SBTTL  DR11-W AST Receiver
          0859   ;++
          0859   ; FUNCTIONAL DESCRIPTION:
          0859           This routine will be called when DR11-W I/O completes for those $QIOs
          0859   ;       which specify an ASTADR parameter.
          0859   ;
          0859   ; CALLING SEQUENCE:
          0859   ;       Called via AST at I/O completion.
          0859   ;
          0859   ; INPUT PARAMETERS:
          0859   ;       NONE
          0859   ;
          0859   ; IMPLICIT INPUTS:
          0859   ;       NONE
          0859   ;
          0859   ; OUTPUT PARAMETERS:
          0859   ;       NONE
          0859   ;
          0859   ; IMPLICIT OUTPUTS:
          0859   ;       NONE
          0859   ;
          0859   ; COMPLETION CODES:
          0859   ;       NONE
          0859   ;
          0859   ; SIDE EFFECTS:
          0859   ;       NONE
          0859   ;
          0859   ;--
          0859
          0859   IOAST:
    0FFC  0859           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
          085B
      04  085B           RET
```

```
                              085C        .SBTTL  Restore Original DR11-W Characteristics
                              085C  ;++
                              085C  ; FUNCTIONAL DESCRIPTION:
                              085C  ;       This routine will be called when the sequence of $QIOs completes for
                              085C  ;       all DR11-W's or when something causes the test to abort.
                              085C  ;
                              085C  ; CALLING SEQUENCE:
                              085C  ;       BSBW    RESET_DR11WS
                              085C  ;
                              085C  ; INPUT PARAMETERS:
                              085C  ;       NONE
                              085C  ;
                              085C  ; IMPLICIT INPUTS:
                              085C  ;       Original characteristics buffer for each DR11-W.  We will assume there
                              085C  ;       are valid data if there is anything in here.
                              085C  ;
                              085C  ; OUTPUT PARAMETERS:
                              085C  ;       NONE
                              085C  ;
                              085C  ; IMPLICIT OUTPUTS:
                              085C  ;       NONE
                              085C  ;
                              085C  ; COMPLETION CODES:
                              085C  ;       NONE
                              085C  ;
                              085C  ; SIDE EFFECTS:
                              085C  ;       Each DR11-W has the same characteristics as it had at the start of the
                              085C  ;       test.
                              085C  ;
                              085C  ;--
                              085C
                              085C  RESET_DR11WS:
                              085C
     56    0300'CF      DE    085C          MOVAL   UNIT_LIST,R6                    ; This will point to current node
                              0861  10$:
          56    66      C0    0861          ADDL2   (R6),R6                         ; Point to the next node
  56   00000300'8F      D1    0864          CMPL    #UNIT_LIST,R6                   ; Back at the queue head?
                43      13    086B          BEQL    20$                             ; BR if so
           01B4 C6      D5    086D          TSTL    XA_Q_ORIGINAL(R6)               ; Did we ever get DR11-W characs?
                EE      13    0871          BEQL    10$                             ; BR if not
     5A    01BC C6      DE    0873          MOVAL   XA_K_QIO(R6),R10                ; Point to $QIO arglist for clarity
             28 AA      7C    0878          CLRQ    QIO$_P4(R10)                    ; Clear P4 and P5
             24 AA      D4    087B          CLRL    QIO$_P3(R10)                    ; Clear P3
        20 AA   04      D0    087E          MOVL    #4,QIO$_P2(R10)                 ; We must transfer at least some data
   1C AA   09F2 C6      DE    0882          MOVAL   XA_K_BUF(R6),QIO$_P1(R10) ; P1 gets data buffer
             14 AA      7C    0888          CLRQ    QIO$_ASTADR(R10)                ; Clear ASTADR and ASTPRM
OC AA   0000190B 8F     D0    088B          MOVL    #XAW__RESET_CYCLE,QIO$_FUNC(R10) ; Set FUNC
                              0893          $QIOW_G (R10)                           ; Reset DR11-W
             20 AA      D4    089A          CLRL    QIO$_P2(R10)                    ; Clear P2
   1C AA   01B4 C6      DE    089D          MOVAL   XA_Q_ORIGINAL(R6),QIO$_P1(R10) ; P1 gets characteristics buffer
        0C AA   1A      D0    08A3          MOVL    #IO$_SETCHAR,QIO$_FUNC(R10) ; Set FUNC.  Other args must be OK
                              08A7          $QIOW_G (R10)                           ; Reset original DR11-W characteristics
                              08AE          ; We don't care about failure of these two $QIOs.  There's nothing we
                              08AE          ; can do to remedy the situation now, anyway.
                B1      11    08AE          BRB     10$                             ; Loop to next unit
                              08B0  20$:
                        05    08B0          RSB
```

```
                              08B1   1250               .SBTTL  Timer Expiration Routine
                              08B1   1251   ;++
                              08B1   1252   ; FUNCTIONAL DESCRIPTION:
                              08B1   1253   ;        This routine will be called if the normal three-minute timer to
                              08B1   1254   ;        indicate the end of the test goes off.
                              08B1   1255   ;
                              08B1   1256   ; CALLING SEQUENCE:
                              08B1   1257   ;        Called via AST at $SETIMR expiration.
                              08B1   1258   ;
                              08B1   1259   ; INPUT PARAMETERS:
                              08B1   1260   ;        NONE
                              08B1   1261   ;
                              08B1   1262   ; IMPLICIT INPUTS:
                              08B1   1263   ;        NONE
                              08B1   1264   ;
                              08B1   1265   ; OUTPUT PARAMETERS:
                              08B1   1266   ;        NONE
                              08B1   1267   ;
                              08B1   1268   ; IMPLICIT OUTPUTS:
                              08B1   1269   ;        NONE
                              08B1   1270   ;
                              08B1   1271   ; COMPLETION CODES:
                              08B1   1272   ;        NONE
                              08B1   1273   ;
                              08B1   1274   ; SIDE EFFECTS:
                              08B1   1275   ;        Sets a flag to indicate timer expiration.
                              08B1   1276   ;
                              08B1   1277   ;--
                              08B1   1278
                              08B1   1279   TIME_OUT:
                       0FFC   08B1   1280           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                              08B3   1281
      0002'CF    02      A8   08B3   1282           BISW2   #TEST_OVERM,FLAG          ; Indicate end of pass or of test
                         04   08B8   1283           RET
```

```
08B9          .SBTTL  System Service Exception Handler
08B9   ;++
08B9   ; FUNCTIONAL DESCRIPTION:
08B9   ;       This routine is executed if a software or hardware exception occurs or
08B9   ;       if a LIB$SIGNAL system service is used to output a message.
08B9   ;
08B9   ; CALLING SEQUENCE:
08B9   ;       Entered via an exception from the system
08B9   ;
08B9   ; INPUT PARAMETERS:
08B9   ;       ERROR_COUNT    = previous cumulative error count
08B9   ;                        ------------------
08B9   ;          AP ---->      !       2        !
08B9   ;                        !----------------!
08B9   ;                        ! SIGNL ARY PNT  !
08B9   ;                        !----------------!
08B9   ;                        ! MECH  ARY PNT  !
08B9   ;                        !----------------!  ---------
08B9   ;                        !       4        !          ^
08B9   ;                        !----------------!          !
08B9   ;                        ! ESTABLISH FP   !          !
08B9   ;                        !----------------!          !
08B9   ;                        !     DEPTH      !  Mechanism Array
08B9   ;                        !----------------!          !
08B9   ;                        !       R0       !          !
08B9   ;                        !----------------!          !
08B9   ;                        !       R1       !          v
08B9   ;                        !----------------!  ---------
08B9   ;                        !       N        !          ^
08B9   ;                        !----------------!          !
08B9   ;                        ! CONDITION NAME !          !
08B9   ;                        !----------------!          !
08B9   ;                        ! N-3 ADDITIONAL !     Signal Array
08B9   ;                        ! LONG WORD ARGS !          !
08B9   ;                        !----------------!          !
08B9   ;                        !       PC       !          !
08B9   ;                        !----------------!          !
08B9   ;                        !      PSL       !          v
08B9   ;                        !----------------!  ---------
08B9   ; IMPLICIT INPUTS:
08B9   ;       NONE
08B9   ;
08B9   ; OUTPUT PARAMETERS:
08B9   ;       NONE
08B9   ;
08B9   ; IMPLICIT OUTPUTS:
08B9   ;       NONE
08B9   ;
08B9   ; COMPLETION CODES:
08B9   ;       SS$_NORMAL if it's a UETP condition or RMS error.
08B9   ;       Error status from exception, otherwise.
08B9   ;
08B9   ; SIDE EFFECTS:
08B9   ;       May branch to ERROR_EXIT.
08B9   ;       May print a message.
08B9   ;--
08B9
```

```
UET
Pse

PSE
---
.
$AE
DEV
ROD
RWD
$RM
DR1

Pha
---
Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
135
The
169
67

Mac
---
_$2
_$2
_$2
TOT

207

The

MAC
```

```
                         08B9  SSERROR:
                 0FFC    08B9          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                         08BB
                         08BB          $SETAST_S ENBFLG = #0            ; Disable AST delivery
               01   DD   08C4          PUSHL   #1                       ; Assume ASTs were enabled
         50    09   D1   08C6          CMPL    S^#SS$_WASSET,R0         ; Were ASTs enabled?
               02   13   08C9          BEQL    10$                      ; BR if they were
               6E   D4   08CB          CLRL    (SP)                     ; Set ASTs to remain disabled
                         08CD  10$:
                         08CD          $SETSFM_S ENBFLG = #0            ; Disable SS failure mode
               01   DD   08D6          PUSHL   #1                       ; Assume SS failure mode was enabled
         50    09   D1   08D8          CMPL    S^#SS$_WASSET,R0         ; Was SS failure mode enabled?
               02   13   08DB          BEQL    20$                      ; BR if it was
               6E   D4   08DD          CLRL    (SP)                     ; Set SS failure mode to remain off
                         08DF  20$:
      56    04 AC   D0   08DF          MOVL    CHF$L_SIGARGLST(AP),R6   ; Get the signal array pointer
      59    04 A6   7D   08E3          MOVQ    CHF$L_SIG_NAME(R6),R9    ; Get NAME in R9 and ARG1 in R10
               10   ED   08E7          CMPZV   #STS$V_FAC_NO,-          ; Is this a message from LIB$SIGNAL?
               0C        08E9                  #STS$S_FAC_NO,-
 00000074 8F   59        08EA                  R9,#UETP$_FACILITY
               14   12   08F0          BNEQ    30$                      ; BR if this is not a UETP exception
         66    02   C2   08F2          SUBL2   #2,CHF$L_SIG_ARGS(R6)    ; Drop the PC and PSL
                         08F5          $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
               21   11   0904          BRB     40$                      ; Restore ASTs and SS fail mode
                         0906  30$:
 59  0000045C 8F    D1   0906          CMPL    #SS$_SSFAIL,R9           ; RMS failures are SysSvc failures
               32   12   090D          BNEQ    50$                      ; BR if this can't be an RMS failure
               10   ED   090F          CMPZV   #STS$V_FAC_NO,-          ; Is it an RMS failure?
               0C        0911                  #STS$S_FAC_NO,-
         01    5A        0912                  R10,#RMS$_FACILITY
               2B   12   0914          BNEQ    50$                      ; BR if not
 5A  F0000000 8F    CA   0916          BICL2   #^XF0000000,R10          ; Strip control bits from status code
      08 A6    04   39   091D          MATCHC  #4,CHF$L_SIG_ARG1(R6),-  ; Is it an RMS failure for which...
               14        0921                  #NRAT_LENGTH,-
        004D'CF          0922                  NO_RMS_AST_TABLE         ; ...no AST can be delivered?
               1A   13   0925          BEQL    50$                      ; BR if so - must give error here
                         0927  40$:
               01   BA   0927          POPR    #^M<R0>                  ; Restore SS failure mode...
                         0929          $SETSFM_S ENBFLG = R0            ; ...
               01   BA   0932          POPR    #^M<R0>                  ; Restore AST enable...
                         0934          $SETAST_S ENBFLG = R0            ; ...
         50    01   D0   093D          MOVL    S^#SS$_NORMAL,R0         ; Supply a standard status for exit
                    04   0940          RET                              ; Resume processing (or goto RMS_ERROR)
                         0941  50$:
     02B6'CF   59   D0   0941          MOVL    R9,STATUS                ; Save the status
               58   D4   0946          CLRL    R8                       ; Assume for now it's not SS failure
 59  0000045C 8F    D1   0948          CMPL    #SS$_SSFAIL,R9           ; But is it a System Service failure?
               38   12   094F          BNEQ    70$                      ; BR if not - no special case message
                         0951          $GETMSG_S MSGID = R10,-          ; Get SS failure code associated text
                         0951                    MSGLEN = BUFFER_PTR,-
                         0951                    BUFADR = FAO_BUF,-
                         0951                    FLAGS  = #14,-
                         0951                    OUTADR = MSG_BLOCK
     02E5'CF   95        0968          TSTB    MSG_BLOCK+1              ; Get FAO arg count for SS failure code
               16   13   096C          BEQL    60$                      ; Don't use $GETMSG if no $FAO args...
     000C'CF   DF        096E          PUSHAL  BUFFER_PTR               ; ...else build up...
               01   DD   0972          PUSHL   #1                       ; ...a message describing...
        00741130 8F   DD  0974            PUSHL   #UETP$_TEXT                   ; ...why the System Service failed
              00 5A   F0  097A            INSV    R10,#STS$V_SEVERITY,-         ; Give the message...
              6E 03       097D                    #STS$S_SEVERITY,(SP)          ; ...the correct severity code
              58 03   D0  097F            MOVL    #3,R8                         ; Count the number of args we pushed
              05      11  0982            BRB     70$
                          0984       60$:
                 5A   DD  0984            PUSHL   R10                           ; Save SS failure code
              58 01   D0  0986            MOVL    #1,R8                         ; Count the number of args we pushed
                          0989       70$:
        57    66 04   C5  0989            MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
              5E 57   C2  098D            SUBL2   R7,SP                            ; Save the current signal array...
     6E    04 A6 57   28  0990            MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
        7E    66 58   C1  0995            ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
            00A6      31  0999            BRW     ERROR_EXIT
```

```
                           099C  1414          .SBTTL  RMS Error Handler
                           099C  1415  ;++
                           099C  1416  ; FUNCTIONAL DESCRIPTION:
                           099C  1417  ;       This routine handles error returns from RMS calls.
                           099C  1418  ;
                           099C  1419  ; CALLING SEQUENCE:
                           099C  1420  ;       Called by RMS when a file processing error is found.
                           099C  1421  ;
                           099C  1422  ; INPUT PARAMETERS:
                           099C  1423  ;       The FAB or RAB associated with the RMS call.
                           099C  1424  ;
                           099C  1425  ; IMPLICIT INPUTS:
                           099C  1426  ;       NONE
                           099C  1427  ;
                           099C  1428  ; OUTPUT PARAMETERS:
                           099C  1429  ;       NONE
                           099C  1430  ;
                           099C  1431  ; IMPLICIT OUTPUTS:
                           099C  1432  ;       Error message
                           099C  1433  ;
                           099C  1434  ; COMPLETION CODES:
                           099C  1435  ;       NONE
                           099C  1436  ;
                           099C  1437  ; SIDE EFFECTS:
                           099C  1438  ;       Program may exit, depending on severity of the error.
                           099C  1439  ;
                           099C  1440  ;--
                           099C  1441
                           099C  1442  RMS_ERROR:
                     0FFC  099C  1443          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                           099E  1444
        56   04 AC    D0   099E  1445          MOVL    4(AP),R6                  ; See whether we're dealing with...
           66   03    91   09A2  1446          CMPB    #FAB$C_BID,FAB$B_BID(R6)  ; ...a FAB or a RAB
                16    12   09A5  1447          BNEQ    10$                       ; BR if it's a RAB
        57  01F5'CF   DE   09A7  1448          MOVAL   FILE,R7                   ; FAB-specific code:  text string...
           58   56    D0   09AC  1449          MOVL    R6,R8                     ; ...address of FAB...
             0C A6    DD   09AF  1450          PUSHL   FAB$L_STV(R6)             ; ...STV field for error...
             08 A6    DD   09B2  1451          PUSHL   FAB$L_STS(R6)             ; ...STS field for error...
   02B6'CF   08 A6    D0   09B5  1452          MOVL    FAB$L_STS(R6),STATUS      ; ...and save the error code
                15    11   09BB  1453          BRB     COMMON                    ; FAB and RAB share other code
                           09BD  1454  10$:
        57  0201'CF   DE   09BD  1455          MOVAL   RECORD,R7                 ; RAB-specific code:  text string...
        58   3C A6    D0   09C2  1456          MOVL    RAB$L_FAB(R6),R8          ; ...address of associated FAB...
             0C A6    DD   09C6  1457          PUSHL   RAB$L_STV(R6)             ; ...STV field for error...
             08 A6    DD   09C9  1458          PUSHL   RAB$L_STS(R6)             ; ...STS field for error...
   02B6'CF   08 A6    D0   09CC  1459          MOVL    RAB$L_STS(R6),STATUS      ; ...and save the error code
                           09D2  1460  COMMON:
        5A   34 A8    9A   09D2  1461          MOVZBL  FAB$B_FNS(R8),R10
                           09D6  1462          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                           09D6  1463                  OUTLEN = BUFFER_PTR,-
                           09D6  1464                  OUTBUF = FAO_BUF,-
                           09D6  1465                  P1     = R7,-
                           09D6  1466                  P2     = R10,-
                           09D6  1467                  P3     = FAB$L_FNA(R8)
        000C'CF       DF   09F0  1468          PUSHAL  BUFFER_PTR                ; ...and arguments for ERROR_EXIT...
             01       DD   09F4  1469          PUSHL   #1                        ; ...
    00741130 8F       DD   09F6  1470          PUSHL   #UETP$_TEXT               ; ...
```

```
                  00   EF  09FC  1471          EXTZV   #STS$V_SEVERITY,-
                  03       09FE  1472                  #STS$S_SEVERITY,-
        59   02B6'CF       09FF  1473                  STATUS,R9                       ; ...get the severity code...
             6E   59   88  0A03  1474          BISB2   R9,(SP)                         ; ...and add it into the signal name
                  05   DD  0A06  1475          PUSHL   #5                              ; Current arg count
                0037   31  0A08  1476          BRW     ERROR_EXIT
```

```
                                  0A0B  1478          .SBTTL  CTRL/C Handler
                                  0A0B  1479 ;++
                                  0A0B  1480 ; FUNCTIONAL DESCRIPTION:
                                  0A0B  1481 ;        This routine handles CTRL/C AST's
                                  0A0B  1482 ;
                                  0A0B  1483 ; CALLING SEQUENCE:
                                  0A0B  1484 ;        Called via AST
                                  0A0B  1485 ;
                                  0A0B  1486 ; INPUT PARAMETERS:
                                  0A0B  1487 ;        NONE
                                  0A0B  1488 ;
                                  0A0B  1489 ; IMPLICIT INPUTS:
                                  0A0B  1490 ;        NONE
                                  0A0B  1491 ;
                                  0A0B  1492 ; OUTPUT PARAMETERS:
                                  0A0B  1493 ;        NONE
                                  0A0B  1494 ;
                                  0A0B  1495 ; IMPLICIT OUTPUTS:
                                  0A0B  1496 ;        NONE
                                  0A0B  1497 ;
                                  0A0B  1498 ; COMPLETION CODES:
                                  0A0B  1499 ;        NONE
                                  0A0B  1500 ;
                                  0A0B  1501 ; SIDE EFFECTS:
                                  0A0B  1502 ;        NONE
                                  0A0B  1503 ;
                                  0A0B  1504 ;--
                                  0A0B  1505
                                  0A0B  1506 CCASTHAND:
                            0FFC  0A0B  1507          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                  0A0D  1508
              00A3'CF         DF  0A0D  1509          PUSHAL  CNTRLCMSG                   ; Set message pointer
                   01         DD  0A11  1510          PUSHL   #1                          ; Set arg count
          00741130 8F         DD  0A13  1511          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                   00         DD  0A19  1512          PUSHL   #0                          ; Indicate an abnormal termination
              0210'CF         DF  0A1B  1513          PUSHAL  PROCESS_NAME                ; ...
                   02         DD  0A1F  1514          PUSHL   #2                          ; ...
          007410E0 8F         DD  0A21  1515          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
   00000000'GF     07         FB  0A27  1516          CALLS   #7,G^LIB$SIGNAL             ; Output the message
                              D0  0A2E  1517          MOVL    #<STS$M_INHIB_MSG!-         ; Set the exit status
                                  0A2F  1518                  SS$_CONTROLC-=
                                  0A2F  1519                  STS$K_SUCCESS+STS$K_WARNING>,-
02B6'CF    10000650 8F            0A2F  1520                  STATUS
                                  0A37  1521          $EXIT_S STATUS                      ; Terminate program cleanly
                                  0A42          .SBTTL  Error Exit
                                  0A42  ;++
                                  0A42  ; FUNCTIONAL DESCRIPTION:
                                  0A42  ;       This routine prints an error message and exits.
                                  0A42  ;
                                  0A42  ; CALLING SEQUENCE:
                                  0A42  ;       MOVx  error status value,STATUS
                                  0A42  ;       PUSHx error specific information on the stack
                                  0A42  ;       PUSHL current argument count
                                  0A42  ;       BRW   ERROR_EXIT
                                  0A42  ;
                                  0A42  ; INPUT PARAMETERS:
                                  0A42  ;       Arguments to LIB$SIGNAL, as above
                                  0A42  ;
                                  0A42  ; IMPLICIT INPUTS:
                                  0A42  ;       NONE
                                  0A42  ;
                                  0A42  ; OUTPUT PARAMETERS:
                                  0A42  ;       Message to SYS$OUTPUT and SYS$ERROR
                                  0A42  ;
                                  0A42  ; IMPLICIT OUTPUTS:
                                  0A42  ;       Program exit
                                  0A42  ;
                                  0A42  ; COMPLETION CODES:
                                  0A42  ;       NONE
                                  0A42  ;
                                  0A42  ; SIDE EFFECTS:
                                  0A42  ;       NONE
                                  0A42  ;
                                  0A42  ;--
                                  0A42
                                  0A42  ERROR_EXIT:
                                  0A42
                                  0A42          $SETAST_S ENBFLG = #0           ; ASTs can play havoc with messages
     15 0002'CF      03      E0   0A4B          BBS     #BEGIN_MSGV,FLAG,10$    ; BR if "begin" msg already printed
                     7E      D4   0A51          CLRL    -(SP)                   ; Set the time stamp flag
                000F'CF      DF   0A53          PUSHAL  TEST_NAME               ; Set the test name
                     02      DD   0A57          PUSHL   #2                      ; Push the argument count
            00741039 8F      DD   0A59          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
    00000000'GF      04      FB   0A5F          CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
                                  0A66  10$:
     02F8'CF      08 8E      C1   0A66          ADDL3   (SP)+,#8,ARG_COUNT      ; Get total # args, pop partial count
                02B2'CF      D6   0A6C          INCL    ERROR_COUNT             ; Keep running error count
                     00      DD   0A70          PUSHL   #0                      ; Push the time parameter
                0210'CF      DF   0A72          PUSHAL  PROCESS_NAME            ; Push test name...
            000F0002 8F      DD   0A76          PUSHL   #^XF0002                ; ...arg count...
            007410E2 8F      DD   0A7C          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
                02B2'CF      DD   0A82          PUSHL   ERROR_COUNT             ; Finish off arg list...
                0210'CF      DF   0A86          PUSHAL  PROCESS_NAME            ; ...
            00010002 8F      DD   0A8A          PUSHL   #^X10002                ; ...
            00748022 8F      DD   0A90          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
    00000000'GF  02F8'CF      FB   0A96          CALLS   ARG_COUNT,G^LIB$SIGNAL  ; Truly bitch
                                  0A9F
                02B6'CF      D5   0A9F          TSTL    STATUS                  ; Did we exit with an error code?
                     09      12   0AA3          BNEQ    20$                     ; BR if we did
            007410E2 8F      D0   0AA5          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
                02B6'CF           0AAB                  STATUS
```

```
                                   0AAE  1580 20$:
        02B6'CF    10000000 8F  C8 0AAE  1581         BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                                   0AB7  1582         $EXIT_S STATUS                  ; Exit in error
```

```
                                 0AC2            .SBTTL  Exit Handler
                                 0AC2  ;++
                                 0AC2  ; FUNCTIONAL DESCRIPTION:
                                 0AC2  ;         This routine handles cleanup at exit.  If the MODE logical name is
                                 0AC2  ;         equated to "ONE", the routine will update the test flag in the
                                 0AC2  ;         UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                                 0AC2  ;         UETUNT$B_FLAGS field of the unit block corresponding to a line in the
                                 0AC2  ;         file.
                                 0AC2  ;
                                 0AC2  ; CALLING SEQUENCE:
                                 0AC2  ;         Invoked automatically by $EXIT System Service.
                                 0AC2  ;
                                 0AC2  ; INPUT PARAMETERS:
                                 0AC2  ;         STATUS  contains the exit status.
                                 0AC2  ;         FLAG    has synchronizing bits.
                                 0AC2  ;         DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV
                                 0AC2  ;
                                 0AC2  ; IMPLICIT INPUTS:
                                 0AC2  ;         UNIT_LIST points to te head of a doubly linked circular list of unit
                                 0AC2  ;                   blocks for the device under test.
                                 0AC2  ;
                                 0AC2  ; OUTPUT PARAMETERS:
                                 0AC2  ;         NONE
                                 0AC2  ;
                                 0AC2  ; IMPLICIT OUTPUTS:
                                 0AC2  ;         Various files are de-accessed, the process name is reset, and any
                                 0AC2  ;         necessary synchronization with UETPDEV01 is carried out.
                                 0AC2  ;         If the MODE logical name is equated to "ONE", the routine will update
                                 0AC2  ;         the test flag in the UETINIDEV.DAT file depending on the
                                 0AC2  ;         UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                                 0AC2  ;         block corresponding to the DR11-W.
                                 0AC2  ;
                                 0AC2  ; COMPLETION CODES:
                                 0AC2  ;         NONE
                                 0AC2  ;
                                 0AC2  ; SIDE EFFECTS:
                                 0AC2  ;         NONE
                                 0AC2  ;
                                 0AC2  ;--
                                 0AC2
                                 0AC2  EXIT_HANDLER:
                       0FFC      0AC2          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 0AC4
                                 0AC4          $SETSFM_S ENBFLG = #0           ; Turn off System Service failure mode
                                 0ACD          $SETAST_S ENBFLG = #0           ; We're finished - no more ASTs
       03 0002'CF    04    E0    0AD6          BBS     #ONE_SHOTV,FLAG,10$     ; If one-shot, update testability...
                   00B8    31    0ADC          BRW     END_UPDATE              ; ...else don't update UETINIDEV.DAT
                                 0ADF  10$:
       03 0002'CF    02    E0    0ADF          BBS     #SAFE_TO_UPDV,FLAG,20$  ; Only update if it's safe
                   00AF    31    0AE5          BRW     END_UPDATE              ; Else forget it
                                 0AE8  20$:
          5A   04BC'CF     DE    0AE8          MOVAL   INI_RAB,R10             ; Set the RAB address
             1E AA   02    90    0AED          MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
   10 AA   0500'CF   06    28    0AF1          MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                                 0AF8          $GET    RAB = (R10)             ; Go back to the DDB record
                75 50    E9    0B01          BLBC    R0,UPDATE_FAILED        ; If failure then forget it
             1E AA   00    90    0B04          MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
```

```
    5B   0300'CF   00000300'8F   C1  0B08              ADDL3    #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
                             59  D4  0B12              CLRL     R9                        ; Init a counter
                                     0B14  UNIT_LOOP:
                             01  E1  0B14              BBC      #UETUNT$V_TESTABLE,-     ; BR if this unit is not testable
                       02 0B AB      0B16                       UETUNT$B_FLAGS(R11),10$
                             59  D6  0B19              INCL     R9                        ; Count testable units
                                     0B1B  10$:
                         5B  6B  C0  0B1B              ADDL2    (R11),R11                 ; Next unit block
             00000300'8F     5B  D1  0B1E              CMPL     R11,#UNIT_LIST            ; Are we full circle in the list?
                             ED  12  0B25              BNEQ     UNIT_LOOP                 ; BR if not
                             59  D5  0B27              TSTL     R9                        ; Any testable units?
                             12  12  0B29              BNEQ     20$                       ; BR if yes...
             0018'CF     4E  8F  90  0B2B              MOVB     #^A/N/,BUFFER+4           ; ...else disable the DDB record...
                                     0B31              $UPDATE  RAB = (R10)               ; ...here
                         3C  50  E9  0B3A              BLBC     R0,UPDATE_FAILED          ; If error then forget it
                                     0B3D  20$:
                         5B  6B  C0  0B3D              ADDL2    (R11),R11                 ; Next unit block
             00000300'8F     5B  D1  0B40              CMPL     R11,#UNIT_LIST            ; Are we full circle in the list?
                             4E  13  0B47              BEQL     END_UPDATE                ; BR if yes
                                     0B49              $GET     RAB = (R10)               ; Get a record
                         24  50  E9  0B52              BLBC     R0,UPDATE_FAILED          ; If error then forget it
                 0014'CF     20  8A  0B55              BICB2    #LC_BITM,BUFFER           ; Convert to uppercase
             0014'CF     55  8F  91  0B5A              CMPB     #^A7U/,BUFFER             ; Is it a UCB record?
                             35  12  0B60              BNEQ     END_UPDATE                ; BR if not
                             01  E0  0B62              BBS      #UETUNT$V_TESTABLE,-     ; BR if this unit is testable...
                       D6 0B AB      0B64                       UETUNT$B_FLAGS(R11),20$
             0018'CF     4E  8F  90  0B67              MOVB     #^A/N/,BUFFER+4           ; ...else disable the UCB record...
                                     0B6D              $UPDATE  RAB = (R10)               ; ...here
                         C4  50  E8  0B76              BLBS     R0,20$                    ; Look at the next record if no error
                                     0B79  UPDATE_FAILED:
                         0C  AA  DD  0B79              PUSHL    RAB$L_STV(R10)            ; Do a simple message...
                             50  DD  0B7C              PUSHL    R0                        ; ...to tell of the failure
                     01B8'CF     DF  0B7E              PUSHAL   INIDEV_UPDERR
                             01  DD  0B82              PUSHL    #1
                             00  EF  0B84              EXTZV    #STS$V_SEVERITY,-        ; Copy the severity from RMS status...
               7E    50      03      0B86                       #STS$S_SEVERITY,R0,-(SP)
           6E   00741130 8F  C8  0B89              BISL2    #UETP$_TEXT,(SP)          ; ...to our message
           00000000'GF   05  FB  0B90              CALLS    #5,G^LIB$SIGNAL
                                     0B97  END_UPDATE:
        03 0002'CF       01  E0  0B97              BBS      #TEST_OVERV,FLAG,10$      ; Did the test complete normally?
                           FCBC  30  0B9D              BSBW     RESET_DR11WS              ; Reset original DR11-W characs if not
                                     0BA0  10$:
                             00  DD  0BA0              PUSHL    #0                        ; Set the time flag
                     000F'CF     DF  0BA2              PUSHAL   TEST_NAME                 ; Push the test name
                             02  DD  0BA6              PUSHL    #2                        ; Push arg count
                             00  EF  0BA8              EXTZV    #STS$V_SEVERITY,-        ; Push the proper exit severity...
                             03      0BAA                       #STS$S_SEVERITY,-
               7E   02B6'CF          0BAB                       STATUS,-(SP)
           6E   00741080 8F  C8  0BAF              BISL2    #UETP$_ENDEDD,(SP)        ; ...and use it in our message code
                             04  DD  0BB6              PUSHL    #4
                         51  5E  D0  0BB8              MOVL     SP,R1
                                     0BBB              $PUTMSG_S MSGVEC = (R1)            ; Output the message
                                     0BCA              $SETPRN_S PRCNAM = ACNT_NAME       ; Reset the process name
                             04      0BD5              RET                                ; That's all folks!
                                     0BD6
                                     0BD6              .END     UETDR1W00
```

```
                                                B  8
UETDR1W00               - VAX/VMS UETP DR11-W EXERCISER       16-SEP-1984 01:25:57  VAX/VMS Macro V04-00        Page  42
Symbol table                                                   5-SEP-1984 04:25:15  [UETP.SRC]UETDR1W00.MAR;1        (24)

$$.TAB            = 00000508 R   04   FAB$B_BID         = 00000000
$$.TABEND         = 00000558 R   04   FAB$B_FNS         = 00000034
$$.TMP            = 00000002          FAB$C_BID         = 00000003
$$.TMP1           = 00000001          FAB$C_BLN         = 00000050
$$.TMP2           = 0000006A          FAB$C_SEQ         = 00000000
$$.TMPX           = 00000016 R   05   FAB$C_VAR         = 00000002
$$.TMPX1          = 0000000D          FAB$L_ALQ         = 00000010
$$ARGS            = 0000000C          FAB$L_DEV         = 00000040
$$T1              = 00000000          FAB$L_FNA         = 0000002C
$$T2              = 00000006          FAB$L_FOP         = 00000004
ACNT_NAME           00000000 R   03   FAB$L_STS         = 00000008
ALL_SET             0000045A R   06   FAB$L_STV         = 0000000C
ARG_COUNT           000002F8 R   04   FAB$V_CHAN_MODE   = 00000002
ASTADR_TABLE        0000057C R   03   FAB$V_CR          = 00000001
BEGIN_MSGM        = 00000008          FAB$V_FILE_MODE   = 00000004
BEGIN_MSGV        = 00000003          FAB$V_GET         = 00000001
BUFFER              00000014 R   04   FAB$V_LNM_MODE    = 00000000
BUFFER_PTR          0000000C R   04   FAB$V_PUT         = 00000000
CCASTHAND           00000A0B R   06   FAB$V_UFO         = 00000011
CHF$L_SIGARGLST   = 00000004          FAB$V_UPD         = 00000003
CHF$L_SIG_ARG1    = 00000008          FAB$V_UPI         = 00000006
CHF$L_SIG_ARGS    = 00000000          FAB$W_GBC         = 00000048
CHF$L_SIG_NAME    = 00000004          FAO_BUF             00000004 R   04
CNTRLCMSG           000000A3 R   03   FILE                000001F5 R   03
COMMON              000009D2 R   06   FIND_IT             0000021C R   06
CONTROLLER          00000031 R   03   FLAG                00000002 R   04
CONT_DESC           000001ED R   03   FOUND_IT            000002B4 R   06
CS1                 00000082 R   03   FUNC_TABLE          00000560 R   03
CS3                 00000094 R   03   ILLEGAL_REC         00000151 R   03
DDB_RFA             00000500 R   04   INADDRESS           000002C2 R   04
DEAD_CTRLNAME       000000E4 R   03   INIDEV_UPDERR       000001B8 R   03
DEBUG_DUMP          00000788 R   06   INI_FAB             0000046C R   04
DEBUG_MSG           000002EF R   03   INI_RAB             000004BC R   04
DEV$V_TRM         = 00000002          INPUT_ITMLST        00000072 R   03
DEVALLOC            00000249 R   03   IO$M_CTRLCAST     = 00000100
DEVDEP_SIZE       = 0000084D          IO$M_CYCLE        = 00001000
DEVDSC              00000208 R   04   IO$M_DIAGNOSTIC   = 00000100
DEVNAM_LEN          000002D2 R   04   IO$M_RESET        = 00000800
DEV_NAME            00000227 R   04   IO$M_SETFNCT      = 00000200
DIAG_BUF            00000310 R   04   IO$M_TIMED        = 00000080
DIAG_MSG            00000444 R   03   IO$_SETCHAR       = 0000001A
DIB                 00000236 R   04   IO$_SETMODE       = 00000023
DIB$B_DEVCLASS    = 00000004          IO$_WRITEPBLK     = 0000000B
DIB$B_DEVTYPE     = 00000005          IOAST               00000859 R   06
DIB$K_LENGTH      = 00000074          ITERATION           000002DC R   04
DIBBUF              0000023E R   04   LC_BITM           = 00000020
DUMMY_QIO           00000529 R   03   LIB$SIGNAL          ********   X 06
DUMP_MODEM        = 00000020          LINE_GEN_COUNT    = 00000007
DUMP_MODEV        = 00000005          MAX_DEV_DESIG     = 0000000A
DVI$_DEVNAM       = 00000020          MAX_PROC_NAME     = 0000000F
DWT_SIZE          = 000003E8          MAX_UNIT_DESIG    = 00000005
EFN2              = 00000004          MINIMUM           = 00000258
END_UPDATE          00000B97 R   06   MODE                00000041 R   03
ERROR_COUNT         000002B2 R   04   MSG_BLOCK           000002E4 R   04
ERROR_EXIT          00000A42 R   06   NAME_LEN          = 0000000F
EXIT_DESC           000002E8 R   04   NEW_NODE            00000308 R   04
EXIT_HANDLER        00000AC2 R   06   NOUNIT_SELECTED     0000012B R   03
```

```
NO_CTRLNAME                     000000C4 R      03        RESET_DR11WS                    0000085C R      06
NO_MESSAGEM                   = 00000040                  RESTART                         00000499 R      06
NO_MESSAGEV                   = 00000006                  RMS$_BLN                        ********    X   03
NO_RMS_AST_TABLE                0000004D R      03        RMS$_BUSY                       ********    X   03
NRAT_LENGTH                   = 00000014                  RMS$_CDA                        ********    X   03
ONE_SHOTM                     = 00000010                  RMS$_FAB                        ********    X   03
ONE_SHOTV                     = 00000004                  RMS$_FACILITY                 = 00000001
OUTADDRESS                      000002CA R      04        RMS$_RAB                        ********    X   03
P1_TABLE                        00000598 R      03        RMS_ERROR                       0000099C R      06
P2_TABLE                        000005B4 R      03        RMS_ERR_STRING                  0000020F R      03
P3_TABLE                        000005D0 R      03        SAFE_TO_UPDM                  = 00000004
P4_TABLE                        000005EC R      03        SAFE_TO_UPDV                  = 00000002
P5_TABLE                        00000608 R      03        SEC$M_EXPREG                    ********    X   06
PAGES                         = 00000007                  SEC$M_GBL                       ********    X   06
PASS                            000002E0 R      04        SHR$_ABENDD                   = 000010E0
PASS_MSG                        00000185 R      03        SHR$_BEGIND                   = 00001038
PMTSIZ                        = 00000019                  SHR$_ENDEDD                   = 00001080
PROCESS_NAME                    00000210 R      04        SHR$_OPENIN                   = 00001098
PROCESS_NAME_FREE             = 0000000B                  SHR$_TEXT                     = 00001130
PROC_CONT_NAME                  0000008B R      06        SLOW_DR11W                      00000272 R      03
PROMPT                          00000230 R      03        SS$_BADPARAM                  = 00000014
QIO$_ASTADR                   = 00000014                  SS$_CONTROLC                  = 00000651
QIO$_ASTPRM                   = 00000018                  SS$_NORMAL                    = 00000001
QIO$_CHAN                     = 00000008                  SS$_NOSUCHSEC                 = 00000978
QIO$_EFN                      = 00000004                  SS$_SSFAIL                    = 0000045C
QIO$_FUNC                     = 0000000C                  SS$_WASSET                    = 00000009
QIO$_IOSB                     = 00000010                  SSERROR                         000008B9 R      06
QIO$_NARGS                    = 0000000C                  SS_SYNCH_EFN                  = 00000003
QIO$_P1                       = 0000001C                  STATUS                          000002B6 R      04
QIO$_P2                       = 00000020                  STR$UPCASE                      ********    X   06
QIO$_P3                       = 00000024                  STS$K_ERROR                   = 00000002
QIO$_P4                       = 00000028                  STS$K_INFO                    = 00000003
QIO$_P5                       = 0000002C                  STS$K_SUCCESS                 = 00000001
QIO$_P6                       = 00000030                  STS$K_WARNING                 = 00000000
QIO_EFN                       = 00000005                  STS$M_INHIB_MSG               = 10000000
QIO_ERROR                       00000297 R      03        STS$S_FAC_NO                  = 0000000C
QIO_FUNC_FAIL                   000002B8 R      03        STS$S_SEVERITY                = 00000003
QUAD_STATUS                     000002BA R      04        STS$V_FAC_NO                  = 00000010
RAB$B_PSZ                     = 00000034                  STS$V_SEVERITY                = 00000000
RAB$B_RAC                     = 0000001E                  SUC_EXIT                        000006BC R      06
RAB$C_BID                     = 00000001                  SUPDEV_GBLSEC                   00000020 R      03
RAB$C_BLN                     = 00000044                  SUP_FAB                         00000508 R      04
RAB$C_RFA                     = 00000002                  SYS$ASSIGN                      ********   GX   06
RAB$C_SEQ                     = 00000000                  SYS$CONNECT                     ********   GX   06
RAB$L_CTX                     = 00000018                  SYS$CRMPSC                      ********   GX   06
RAB$L_FAB                     = 0000003C                  SYS$DCLEXH                      ********   GX   06
RAB$L_PBF                     = 00000030                  SYS$EXIT                        ********   GX   06
RAB$L_ROP                     = 00000004                  SYS$EXPREG                      ********   GX   06
RAB$L_STS                     = 00000008                  SYS$FAO                         ********    X   06
RAB$L_STV                     = 0000000C                  SYS$FAOL                        ********   GX   06
RAB$V_PMT                     = 0000001E                  SYS$GET                         ********   GX   06
RAB$W_RFA                     = 00000010                  SYS$GETDEV                      ********   GX   06
RAB$W_RSZ                     = 00000022                  SYS$GETDVI                      ********   GX   06
RANDOM1                         000002D4 R      04        SYS$GETMSG                      ********   GX   06
RANDOM2                         000002D8 R      04        SYS$INPUT                       00000061 R      03
RECORD                          00000201 R      03        SYS$MGBLSC                      ********   GX   06
REC_SIZE                      = 00000028                  SYS$OPEN                        ********   GX   06
```

```
SYS$PUTMSG                ********* GX    06          XA_Q_IOSB                        000001A4
SYS$QIOW                  ********* GX    06          XA_Q_ORIGINAL                    000001B4
SYS$SETAST                ********* GX    06
SYS$SETIMR                ********* GX    06
SYS$SETPRN                ********* GX    06
SYS$SETSFM                ********* GX    06
SYS$TRNLOG                ********* GX    06
SYS$UPDATE                ********* GX    06
SYSIN_FAB                 000003D8 R      04
SYSIN_RAB                 00000428 R      04
TEST_LOOP                 0000049E R      06
TEST_NAME                 0000000F R      03
TEST_OVERM              = 00000002
TEST_OVERV              = 00000001
TEXT_BUFFER             = 000001F4
THREEMIN                  000001DD R      03
TIME_IT                   00000486 R      06
TIME_OUT                  000008B1 R      06
TIME_STAMP_LEN          = 00000100
TTCHAN                    00000000 R      04
UETDR1W00                 00000000 RG     06
UETP                    = 00740000
UETP$_ABENDD            = 007410E0
UETP$_ABORTC            = 0074832B
UETP$_BEGIND            = 00741038
UETP$_DATADEVERR        = 00748018
UETP$_DENOSU            = 00748333
UETP$_ENDEDD            = 00741080
UETP$_ERBOXPROC         = 00748020
UETP$_FACILITY          = 00000074
UETP$_OPENIN            = 00741098
UETP$_TEXT              = 00741130
UETUNT$B_FLAGS          = 0000000B
UETUNT$B_TYPE           = 00000008
UETUNT$C_DEVDEP         = 000001A4
UETUNT$C_INDSIZ         = 000001A4
UETUNT$L_ITER           = 00000010
UETUNT$M_TESTABLE       = 00000002
UETUNT$T_FILSPC         = 00000014
UETUNT$V_TESTABLE       = 00000001
UETUNT$W_CHAN           = 0000000C
UETUNT$W_SIZE           = 00000009
UNIT_DESC                 000001E5 R      03
UNIT_LIST                 00000300 R      04
UNIT_LOOP                 00000B14 R      06
UPDATE_FAILED             00000B79 R      06
WRITE_SIZE              = 000003E8
XA$M_LINK               = 00000002
XAW__CYCLE              = 0000110B
XAW__RESET_CYCLE        = 0000190B
XAW__SETFNCT_CYCLE      = 0000130B
XAW__TIMED_CYCLE        = 0000118B
XA_B_TSI                  000001F0
XA_K_BUF                  000009F2
XA_K_QIO                  000001BC
XA_K_TSTAMP               000001F1
XA_Q_CHARAC               000001AC
```

```
                                                      +----------------+
                                                      ! Psect synopsis !
                                                      +----------------+

PSECT name                      Allocation           PSECT No.  Attributes
----------                      ----------           ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (   0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)  01 (   1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
DEVDEP_STR_DEF                  00000DDA  (  3546.)  02 (   2.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE   RD   NOWRT NOVEC PAGE
RODATA                          00000624  (  1572.)  03 (   3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC PAGE
RWDATA                          00000558  (  1368.)  04 (   4.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD     WRT NOVEC PAGE
$RMSNAM                         00000023  (    35.)  05 (   5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
DR11W                           00000BD6  (  3030.)  06 (   6.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE

                                                   +------------------------+
                                                   ! Performance indicators !
                                                   +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     32    00:00:00.07    00:00:00.63
Command processing                131    00:00:00.77    00:00:05.53
Pass 1                            723    00:00:24.42    00:00:55.59
Symbol table sort                   0    00:00:02.57    00:00:05.04
Pass 2                            412    00:00:06.00    00:00:11.24
Symbol table output                35    00:00:00.27    00:00:00.74
Psect synopsis output               5    00:00:00.05    00:00:00.08
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             1342    00:00:34.16    00:01:18.85

The working set limit was 2000 pages.
135028 bytes (264 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1724 non-local and 49 local symbols.
1696 source lines were read in Pass 1, producing 38 object records in Pass 2.
67 pages of virtual memory were used to define 56 macros.

                                                  +--------------------------+
                                                  ! Macro library statistics !
                                                  +--------------------------+

Macro library name                                Macros defined
------------------                                --------------
_$255$DUA28:[UETP.OBJ]UETP.MLB;1                              2
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                                0
_$255$DUA28:[SYSLIB]STARLET.MLB;2                            49
TOTALS (all libraries)                                       51

2074 GETS were required to define 51 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:UETDR1W00/OBJ=OBJ$:UETDR1W00 MSRC$:UETDR1W00/UPDATE=(ENH$:UETDR1W00)+EXECML$/LIB+LIB$:UETP/LIB
```

0411 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

UETFORT03
LIS

UETFORT01
LIS

UETLPAK00
LIS

UETNETS00
LIS

UETDR1W00
LIS

UETFORT02
LIS